1991

Fundamentals of manipulator calibration

Mooring, Benjamin W.

Wiley-interscience

FUNDAMENTALS OF MANIPULATOR CALIBRATION

BENJAMIN W. MOORING
IBM Corporation
Austin, Texas

ZVI S. ROTH
Florida Atlantic University
Boca Raton, Florida

MORRIS R. DRIELS
The Naval Postgraduate School
Montery, California

A Wiley-Interscience Publication
JOHN WILEY & SONS, INC.
New York / Chichester / Brisbane / Toronto / Singapore

Library of Congress Cataloging in Publication Data:

Mooring, Benjamin.

Fundamentals of manipulator calibration / Benjamin Mooring, Zvi Roth, Morris Driels.

p. cm.

Includes bibliographical references and index.

ISBN 0-471-50864-0

1. Manipulators (Mechanism)—Calibration. 2. Robots, Industrial-Calibration. I. Roth, Zvi. II. Driels, Morris. III. Title.

TJ211.M65 1991

670.42'72—dc20 90-42916

CIP

Printed in the United of America

10 9 8 7 6 5 4 3 2 1

PREFACE

The purpose of this book is to bring together the many aspects of robot calibration from the arenas of industrial application and academic research, and present a structured overview of the topic that can then be of use to engineers in both groups. In attempting to address the interests of both groups, it is hoped that a sound theoretical background is developed for what is, in essence, a very practical problem facing those involved in the implementation of advanced automation. Before outlining individual chapters in this book, it is worth reviewing the short history of robot calibration so that future developments may be observed in this context.

In the years since robot manipulators were first used to automate pick and place operations, it was assumed that different machines of the same model line were identical in their operation, and faulty units could be replaced without consideration to whether the new unit would behave exactly as its predecessor. Engineers found this assumption to be incorrect and the new units had to be manually retaught to perform the required operation. Apparently this procedure was acceptable in most cases since machine breakdowns were rare and the number of locations taught to the arm was small. The concept of off-line programming was developed as a means of automatically generating robot control programs for otherwise tedious applications involving large numbers of taught points. For tasks such as the insertion of large numbers of electronic components into a printed circuit card, the existing data base used to manufacture the card was used to define the locations of the insertions, thereby avoiding the time-consuming and error-prone manual teaching phase. In transforming the insertion locations, as defined by the task, to sets of joint angles, as used by the robot, the ideal, or nominal, kinematic model of the arm was used. These transformed task points were used to generate the program that moved the arm sequentially

from one point to the other. When completed, engineers noticed substantial differences between where the arm was supposed to go and where the program sent it. In addition, the errors were so large that many practical tasks could not be performed solely by off-line programming techniques.

As a result of such observed behavior, the concepts of repeatability and accuracy as applied to manipulators became apparent. With these concepts came the realization that even small deviations from the nominal robot geometry could produce substantial errors at the end effector. For example, an angular error of only 0.5° at a rotary joint will produce an error at the end of a 6-ft arm of over 0.5 in. This problem was well known to the U.S. machine tool industry, which had realized that overseas competitors were able to produce more accurate machines at a lower price. This was achieved not by manufacturing tools that were better engineered with tighter tolerances and stiffer mechanical structures, but by building the machine to a high level of mechanical repeatability, and compensating for absolute accuracy deficiencies with internal compensation software on an individual machine basis. Such a procedure is defined as machine calibration, and is consistent with our intuitive concepts of calibration of measurement instruments such as oscilloscopes or voltmeters.

Machine tool manufacturers realized the economic advantages of calibration and began to look at these techniques for high precision measurement systems, such as coordinate measurement machines. At about that time, robot calibration became a significant problem. Initially the two groups of engineers, one consisting of those working with machine tools and the other working in the newer area of robotics, used different approaches and different mathematical tools. Typically, the two groups were unaware of each others efforts. With the growth of published work in robotics in the early 1980s, however, the gaps were filled and the resources of both groups were brought to bear on what was basically the same problem. Even with this coalescence of ideas, it is still valuable to realize that those working in the calibration of machine tools are generaly involved in orders of magnitude of accuracy greater than those working with industrial manipulators. As a result, methods for calibrating machine tools may involve techniques that are deemed to provide "higher order" improvements for robot calibration at the expense of much complexity. These methods may include effects such as ambient temperature variations or structural loading conditions that may be easily defined for a coordinate measuring machine and difficult to characterize for a robot. It is worthwhile, however, for those interested in robot calibration to keep in touch with what is going on in the world of machine tools.

For the purposes of developing a structure, or procedure, for robot calibration, we have divided the overall task into four subtasks: (1) modeling, (2) measurement, (3) identification, and (4) implementation. Although such a structure will be useful in addressing the many factors that affect the accuracy of positioning, our principal interest is in the effect that kinematics, or geometry, has on the manipulator. The basis for this focus is that in most of the calibration tests performed on manipulators by ourselves and other engineers, it was found that correction of the kinematic errors produced improvements in the accuracy

to the same order of magnitude as the repeatability. This is not to say that kinematic errors are the only source of errors, just that experience has shown them to be the most common.

The introductory chapter defines some of the terminology to be used throughout the book, and gives some examples of how relatively small geometric errors can produce significant positioning errors at the end effector. The source of errors is discussed further as is the methodology for defining the calibration process. Chapter 2 develops the modeling theme by describing some of the techniques used to describe the geometry of a wide range of manipulators using a consistent and easily used convention. Despite some drawbacks for certain types of manipulator structure, the Denavit–Hartenberg method is used to illustrate how kinematic errors propagate through a multilink manipulator, and how the system Jacobian relates the variations in nominal kinematic parameters to a set of measurements of the manipulator pose. This relationship forms the basis of the calibration procedure and is utilized extensively later in the book. This chapter continues by looking at issues of model equivalence and completeness, enabling the work of other researchers to be compared and evaluated. Cases of special kinematic configurations such as closed chains are dealt with as are alternative modeling techniques designed for configurations known to cause difficulties with the Denavit–Hartenberg approach.

The calibration equations developed in Chapter 2 imply that measurements have to be made to find the deviations from the nominal kinematic structure. Chapter 3 addresses the various issues associated with the measurement process. Just as in instrument calibration, the measurement system used should ideally be at least one order of magnitude more accurate than the device being calibrated. This requirement often poses stringent constraints on the type of measurement systems available for robot calibration. The ideal measurement system would be capable of measuring all six components of pose, including three spatial positions and three orientations. No commercially available system capable of doing this with the required accuracy, however, has been demonstrated. Typically, measuring systems obtain only partial pose information. For example, only the three spatial displacements from an arbitrary origin may be provided. The measurement systems described in Chapter 3, therefore, are structured according to the amount of pose data obtained.

Once manipulator pose measurements have been taken, and a suitable mathematical model of the robot has been developed, the deviations of the actual kinematics from the nominal values have to be calculated. This is the focus of Chapter 4, and is known as the identification step. Parameter identification has been achieved in practice using a variety of readily available software packages, such as those contained in the IMSL FORTRAN subroutine libraries. This chapter describes the underlying theory behind these identification methods both for linear and nonlinear approaches to the problem. Linear methods involve least-squares estimation theory, minimum variance estimation, and the application of Kalman filtering techniques to kinematic parameter error estimation. The alternative nonlinear search techniques are simpler since they do not rely on an

analytical expression for the system Jacobian, but use only forward kinematic models, at the expense of increased computation time. A case study of a particular manipulator is provided, which shows the application of these identification techniques, and indicates the kind of performance measures available from each to predict the accuracy of identification. The study also provides insight into the planning of robot calibration experiments by optimizing the location of the points used to take pose measurements of the robot end effector. The influence that the number and location of such points has on the resulting accuracy of identification is also examined.

Chapter 5 is concerned with the implementation of the calibrated robot data. Ideally the robot controller should have the actual kinematic parameters "embedded" in it. This, however, is rarely a practical option. Usually the required off-line locations in the task space are converted to modified locations in the robot joint space through an inverse kinematic analysis using the actual kinematic data. This may pose a problem since a robot that has an analytical solution to the inverse kinematics problem for the nominal model may have no such solution for the kinematic model after calibration. These ideas are developed fully in this chapter.

The work developed so far involving modeling, measurement, identification, and implementation is integrated in Chapter 6 using a case study. The complete calibration of a PUMA manipulator is described in detail and covers all of the above steps. Two kinematic models are used to indicate that the resulting improvement in accuracy is indeed independent of model choice. Measurements are taken in the laboratory with a small coordinate measuring machine. A parameter identification program is written in FORTRAN and uses IMSL routines to perform the identification. A complete listing of the source code for this identification is given in the Appendix. Finally, the improved accuracy of the robot is assessed and shown to be better by more than one order of magnitude.

The book concludes with a short description in Chapter 7 of the current status of robot performance standards. Although not the same as calibration, performance assessment of manipulators is of considerate interest to manufacturers and users alike. Some of the most popular methods of performance measurement are described. It is shown that robot performance measurement shares many of the measurement techniques used in robot calibration. Unlike a typical robot calibration, however, areas other than kinematic compensation are usually addressed. These areas are also outlined in Chapter 7.

Finally, it is our pleasure to acknowledge the contributions of many people and organizations over the years who have shared our involvement with robotics in general and robot calibration in particular. Of particular note is the contribution of our colleague, Dr. Louis Everett. His patient assistance and valuable insights have been of significant benefit to our research efforts. Louis is directly responsible for the unique work on modeling of closed-loop manipulators that is summarized in Chapter 2. More significantly, the results of his efforts are reflected throughout this book. We have also benefited from the contributions by engineers in Brown and Sharpe, General Dynamics (Fort Worth, Texas),

Texas Instruments, LTV Aeroproducts Division, and IBM Corporation (Austin, Texas and Boca Raton, Florida). Research work has been partially supported by Texas A&M University, The Program for Automation in Manufacturing at Texas A&M, and The Florida Atlantic University Robotics Center funded through the Florida High Technology and Industry Council. We would also like to thank the many graduate students who have worked with us in this area. In particular, Dr. Geo-Ry Tang, Dr. Uday Pathre, Dr. Hanqi Zhuang, Mr. Satya Padavala, Mr. Saleem Karimjee, and Mr. Shoupu Chen. We would also like to express our sincere thanks to Mrs. Tammy Spies, Mrs. Patricia Mooring, and Mrs. Joan Buttery for their most capable and patient assistance in preparation of this manuscript.

Benjamin Mooring
Zvi Roth
Morris Driels

Austin, Texas
Boca Raton, Florida
Montery, California
April 1990

CONTENTS

1 OVERVIEW OF MANIPULATOR CALIBRATION 1

1.1 Definitions and Examples / 2
1.2 Relationship between Manipulator Precision and Programming / 11
1.3 Manipulator Calibration / 14
1.3.1 Source and Significance of Manipulator Errors / 15
1.3.2 Levels of Manipulator Calibration / 17
1.3.3 What This Book Addresses / 18
1.4 The Calibration Process / 18
1.4.1 Modeling / 19
1.4.2 Measurement / 19
1.4.3 Identification / 20
1.4.4 Implementation / 20
1.5 Conclusion / 21
References / 21

2 KINEMATIC MODELING FOR ROBOT CALIBRATION 23

2.1 Level 1 Models / 24
2.2 Level 2 Models / 25
2.3 Denavit–Hartenberg Method / 26
2.4 Link Kinematic Error Model / 32
2.5 Manipulator Kinematic Error Model / 35

2.6 Limitations of the Denavit–Hartenberg Method / 38
2.7 Properties of a Good Model / 41
2.8 Model Review / 44
2.8.1 Modifications of the Denavit–Hartenberg Method / 44
2.8.1.1 A Modified Denavit–Hartenberg Model / 45
2.8.1.2 Literature Review / 48
2.8.2 Zero-Reference Model / 49
2.8.3 Single Joint Method / 55
2.8.4 Manipulators with Closed Loops / 56
2.8.4.1 A 4R Mechanism / 58
2.8.4.2 A 5R Mechanism / 59
2.8.4.3 A RRRPR Mechanism / 60
2.8.4.4 An RRSSR Mechanism / 61
2.8.5 Joints Comprised of Higher Pairs / 64
2.9 Conclusion / 66
References / 67

3 MEASUREMENT TECHNIQUES FOR MANIPULATOR CALIBRATION 70

3.1 Joint Displacement Transducers / 71
3.1.1 Potentiometers / 72
3.1.2 Encoders / 73
3.1.3 Resolvers / 77
3.1.4 Less Common Devices / 79
3.2 Various Measurement Technologies / 80
3.2.1 Theodolites / 80
3.2.2 Laser Interferometers / 81
3.2.3 Coordinate Measuring Machines / 83
3.2.4 Time of Flight Devices / 84
3.2.5 Camera-Type Devices / 86
3.2.6 Short-Range Devices / 87
3.3 Measuring Methodologies for Robot Calibration / 90
3.3.1 Single Theodolite / 90
3.3.2 Point Measurement / 92
3.3.3 Partial Pose Measurement / 100
3.3.4 Complete Pose Measurement / 101
3.4 Conclusion / 104
References / 104

4 PARAMETER IDENTIFICATION FOR ROBOT CALIBRATION 106

4.1 Identification Issues / 106
4.2 Mathematical Formulation for Identification of Robot Kinematics / 109
4.3 Linear Least-Squares Parameter Estimation / 114
4.3.1 Standard Linear Least-Squares Estimation / 115
4.3.2 Linear Minimum Variance Estimation / 119
4.3.3 Linear Least-Squares—Practical Considerations / 122
4.3.4 Kalman Filtering / 124
4.4 Nonlinear Least-Squares Estimation / 132
4.4.1 Direct Search Methods / 132
4.4.2 Gradient Methods / 135
4.5 Observation Strategy for Robot Kinematic Identification / 140
4.5.1 Numerical Sensitivity of Least-Squares Identification Algorithms / 141
4.5.2 Identification Study of a 5R1P Manipulator / 144
4.5.3 Observability of Kinematic Parameter Errors / 154
4.5.4 Observation Strategy from a Kalman Filtering View / 157
4.5.5 Two-Link Manipulator Example / 160
4.6 Identification of Robot Joint Axes / 165
4.6.1 Linear Regression Analysis / 167
4.6.2 Fitting Data Points to Planes, Circles, and Lines / 171
4.6.2.1 Planes / 171
4.6.2.2 Circles on a Known Plane / 172
4.6.2.3 Lines in 3D / 174
4.6.3 Circle Point Analysis—The Measurement Phase / 175
4.6.4 Iterative Identification of the Joint Axes / 177
4.6.5 Kinematic Parameter Extraction from Identified Joint Axes / 182
4.6.5.1 Stone's Method for Kinematic Parameter Extraction from Identified Joint Axes / 182
4.6.5.2 Sklar's Method for Kinematic Parameter Extraction from Identified Joint Axes / 186
4.7 Observability Issues in Kinematic Error Parameter Identification of Base and Tool Transformations / 192
4.7.1 Introduction / 192
4.7.2 Right and Left Differential Transformations / 192

4.7.3 Generic Forms of Linearized Kinematic Error Models / 196
4.7.3.1 Linear Mappings Relating Cartesian Errors of an End Effector to Cartesian Errors of Individual Links / 196
4.7.3.2 Linear Mapping Relating Cartesian Errors to Link Parameter Errors / 198
4.7.4 Error Model Irreducibility / 199
4.7.5 Observability of Kinematic Error Parameters / 201
4.7.6 Observability Measures of Reducible Error Models / 204
4.8 Conclusion / 204
References / 205

5 IMPLEMENTATION OF MANIPULATOR CALIBRATION 207

5.1 Accuracy Problems in Taught and Data-Driven Applications / 209
5.2 Compensation Algorithms for Robot Geometric Errors / 215
5.2.1 Inverse Jacobian Based Updating of Joint Commands / 215
5.2.2 Redefinition of Task Points Displacement Matrices / 221
5.3 Optimal Design of Robot Accuracy Compensators / 225
5.3.1 Mathematical Background—Linear Quadratic Regulators for First-Order Discrete Time Systems / 226
5.3.2 Joint Command Updating Problem Formulation as an Optimal Control Problem / 229
5.3.3 Optimal Control Solution to the Calibration Compensation Problem / 231
5.3.4 Calibration for Reduced-Order Error Models / 235
5.3.5 Simulation Example of a PUMA 560 Robot / 240
5.4 Nonparametric Accuracy Compensation / 242
5.4.1 Robot Calibration Using Polynomial Approximating Functions / 243
5.4.2 Cerebellar Model Articulation Controller (CMAC) / 253
5.4.2.1 Description of CMAC / 253
5.4.2.2 CMAC and Learning / 256
5.4.2.3 An Example Using CMAC / 256
5.5 Conclusion / 264
References / 264

6 CASE STUDY 266

6.1 Modeling / 266
6.1.1 Modified Denavit–Hartenberg Model / 266
6.1.2 Zero Reference Position Model / 271

6.2 Measurement / 275
6.3 Identification / 282
6.3.1 Identification Algorithm / 285
6.3.2 Identification Results—Modified DH Model / 288
6.3.3 Identification Results—Zero Reference Position Model / 289
6.3.4 Comparison of Identified Model Geometry / 290
6.4 Assessment of Calibrated Robot / 295
6.5 Conclusion / 296
Reference / 297

7 PERFORMANCE EVALUATION 298

7.1 Performance Standards / 298
7.2 Resolution, Repeatability, and Accuracy / 301
7.3 Path Control / 305
7.4 Speed / 306
7.5 Payload / 306
7.6 Thermal Sensitivity / 307
7.7 Compliance / 309
7.8 Conclusion / 311
References / 311

Appendix A—Parameter Identification Program 313

Appendix B—Subroutines Associated with Modified Denavit–Hartenberg Method 316

Appendix C—Subroutines Associated with Zero Reference Position Method 320

Appendix D—General Purpose Subroutines 323

Index 325

FUNDAMENTALS OF MANIPULATOR CALIBRATION

CHAPTER 1

OVERVIEW OF MANIPULATOR CALIBRATION

In the late 1970s and early 1980s, interest in the application of robot manipulators to automated manufacturing soared. The advent of highly capable computer controlled manipulators seemed to indicate that truly flexible automation was feasible and many manufacturers rushed to take advantage of this technology. Unfortunately, the goal of using a robot manipulator as the key element in a flexible manufacturing system proved to be elusive for many manufacturing tasks. The failure of robotics to live up to initial expectations may be attributed to a number of factors. The high initial capital costs of automation along with other economic and technical problems caused many managers to avoid the use of robots.

Although the use of robot manipulators in flexible manufacturing systems still presents significant problems, the goal remains a highly desirable one. One of the significant technical problems to be addressed is the inability of most robots to be programmed off-line or to share programs with other robots. To attempt to solve this problem, a good deal of attention has been paid to the area of manipulator calibration. In this book, we will attempt to meet two primary objectives. The first will be to demonstrate the need for manipulator calibration and to illustrate the significance that the process can have to various aspects of automated manufacturing. The second objective is to describe the details of the calibration procedure. While accomplishing this, we will attempt to point out the various approaches that have been reported as well as the research issues that remain to be addressed. We will begin this process by defining several necessary terms and giving a simple example of the calibration process.

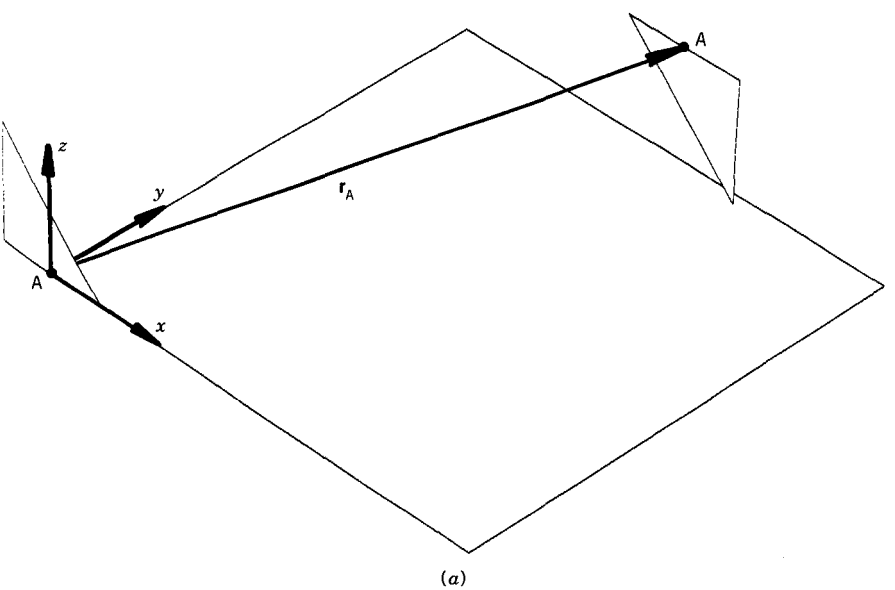
1.1 DEFINITIONS AND EXAMPLES

Typically, a robot manipulator consists of a set of rigid links connected by joints. One end of the manipulator is attached to a rigid surface and is referred to as the *base*. The other end of the manipulator is equipped with a surface that allows the mounting of a specialized gripper or tool, which we will refer to as the *end effector*. The primary purpose of the manipulator is to move the end effector to a specified position or along a specified trajectory. Since the end effector is usually a rigid body being moved in three-dimensional space, it is important that the robot achieve the desired orientation as well as position. The combination of position and orientation will be referred to as a *pose*. For example, assume that an end effector consists of a thin, cylindrical rod. If we wish to position the end of the rod at a given point in space, there are an infinite number of directions from which we may approach the point and still reach the desired position. Although all of these configurations place the end of the rod at the desired position, they each have a different orientation and, hence, a different pose. For a rigid body moving in three-dimensional space, six quantities are required to completely define a pose. The position of a point on the body may be defined by specifying the three position coordinates of the point in some convenient coordinate system. Likewise, three angles may be specified to define the orientation of the body. When specified in this manner, we will say that the pose has been defined in *task space*. Since there is a relationship between the configuration of a manipulator and the pose of the end effector, a pose may also be defined by specifying the geometry of the manipulator and the joint displacements necessary to achieve the pose. When specified in this manner, we will say that the pose has been defined in *joint space*. It is important that the distinction between task space and joint space is clearly understood. To define a pose in task space, a reference coordinate system is established and the position and the orientation of the end effector are specified in this coordinate system. The end effector pose may be described in task space, therefore, without knowledge of the manipulator geometry or configuration. In fact, the task space pose description is completely independent of the manipulator. To describe a pose in joint space, however, one must precisely know the geometry of the manipulator and then specify the joint displacements for that particular pose. As an example, consider the pose illustrated in Figures 1.1a and 1.1b. The end effector in this example is represented by the triangular plate shown in Figure 1.1a. The plate is initially defined at the origin of the task space coordinate system and is then displaced to a new position as shown in the figure. In the task space description, point A on the end effector is located in the reference coordinate system by the vector $\mathbf{r}_A$ that has three Cartesian components in the reference coordinate system. Furthermore, the orientation of the end effector may be specified by the three angles θ_x, θ_y, and θ_z. These angles represent consecutive rotations about the x, y, and z axes of the reference coordinate system. In this example, θ_x and θ_z are zero while θ_y is 180°. For convenience, we may combine the elements of position and orientation into one pose vector, $\mathbf{P}$,

given by

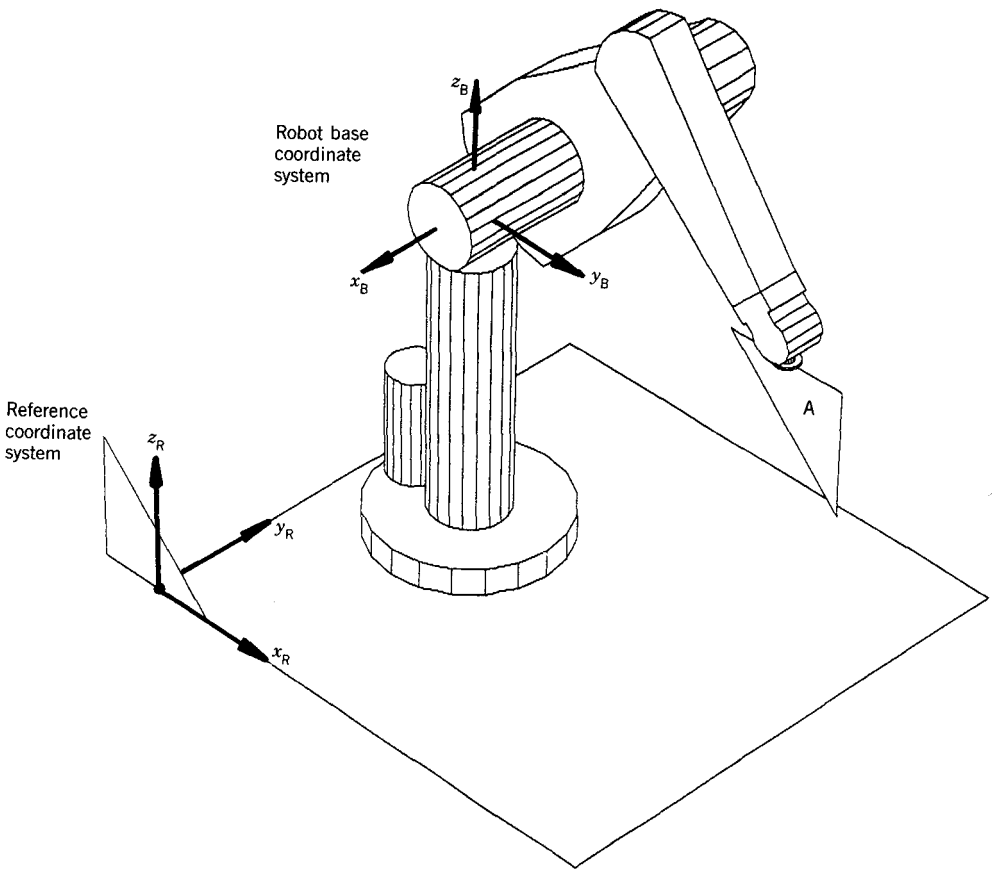
$$\mathbf{P} = \begin{bmatrix} r_x \\ r_y \\ r_z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \tag{1.1}$$

Therefore, the pose vector, **P**, is all that is required to define the pose in task space as illustrated in Figure 1.1a. On the other hand, a joint space description of the same pose requires knowledge of the joint desplacements of the manipulator, the geometry of the manipulator, and the location of the manipulator in the task space coordinate system. This is illustrated in Figure 1.1b. The robot illustrated is a PUMA 560. This information implies that we know the geometry of the manipulator. The location of the robot in the task space is specified by defining the relationship between the robot base coordinate system and the task space coordinate system. It is important to note that for a typical task, the parameters that define the robot geometry and location in the task space will be constants. Once these constants are specified, there is a direct relationship



(a)

Figure 1.1a. Task space pose description.



(*b*)

Figure 1.1b. Joint space pose description.

between the robot configuration as given by the joint displacements and the pose as defined in the task space. This relationship may be expressed as

$$\mathbf{P} = f(\boldsymbol{\eta}, \boldsymbol{\theta}) \tag{1.2}$$

where $\boldsymbol{\eta}$ is a vector of constants that describes the geometry of the manipulator and the location of the manipulator in the workspace and $\boldsymbol{\theta}$ is the vector of joint displacements for any particular pose. Equation 1.2, therefore, represents the transformation from joint space to task space.

Having defined two distinct ways of describing a pose, we will now consider several ways of measuring the ability of a particular manipulator to achieve a given pose. The first and perhaps most common measure of a robot's positioning capability is *repeatability*. Simply put, repeatability is the ability of a manipulator

to return to a previously achieved pose. For example, assume that a manipulator is moved to a particular pose and the associated set of joint displacements is recorded and stored in the robot controller. If the manipulator is moved away from this position and then commanded to move back, the end effector will not return to exactly the same pose. The reasons for this deviation could include small errors in the control of the joint displacements, flexibility in the robot structure, compliance in the joints, or a number of other factors. Repeatability is a measure of the ability of the robot to reachieve the previous pose. It is typically specified as a displacement of the origin of a coordinate system in the end effector after the robot is returned to the specified pose. For example, a repeatability of 0.005 in. would imply that a point on the end effector would always return to within a sphere of radius 0.005 in. This definition of repeatability is easy to understand but extremely difficult to put into practice. For example, the particular point in the end effector that is to be measured is not specified in the definition. If there are significant variations in wrist orientation, a point far from the wrist rotation center would yield a significantly lower repeatability than one close to the center of rotation. Also, no measure of orientation is specified and no indication is given of the location in task space that repeatability is measured. These issues will be addressed at length in Chapter 7, Robot Performance Measures. For the purpose of this discussion, we will use the simple definition of repeatability that is given above.

In addition to repeatability, many robot manufacturers will specify the *resolution* of their robot as a performance measure. Like repeatability, resolution can mean different things to different people. Some engineers think of resolution as the smallest move that a robot can make. This view of resolution becomes difficult to quantify, however, because the minimum possible move may vary significantly throughout the workspace. Another definition for resolution involves the digitization of the various signals moving in and out of the robot controller. Consider a robot joint with a rotary potentiometer as the position feedback transducer. The potentiometer converts the displacement of the joint into a proportional voltage that is used as a feedback signal in the controller. Since robot manipulators are controlled almost exclusively by digital computers, the analog voltage must be converted into a digital value. Let's assume that the analog to digital converter has a resolution of 12 bits. This means that the entire range of the joint must be expressed in 2^{12} or 4096 increments. If the joint range is 360°, any joint motion less than approximately 0.088° cannot be sensed. Throughout the entire robot system there are a number of these digital conversions. The overall resolution of the robot will be given by the coarsest level of digitization for each joint. In other words, the resolution for a joint is the smallest signal change that can be both sensed and acted on by the controller. A more detailed description of resolution is given in Chapter 7, Robot Performance Measures.

Another measure of a manipulator's ability to achieve a specified pose is *accuracy*. Accuracy is the ability of a manipulator to move the end effector to a pose that is specified in task space. The fundamental difference between accuracy and repeatability is that repeatability is the ability of the robot to return to a

previously achieved pose and accuracy is the ability of the manipulator to move to a pose that is specified in task space and that may have never been previously reached. Another way of making the distinction between accuracy and repeatability is to consider the means by which the goal pose is specified. When considering repeatability, the pose has been previously attained and, therefore, the necessary joint displacements are known. In other words, the pose has been specified in joint space. With accuracy, however, the pose is specified in task space and the particular set of joint angles necessary to achieve the pose must be determined. As with repeatability, a more complete description of accuracy measurement will be given in Chapter 7.

Experience has shown that industrial manipulators have much better repeatability than accuracy. The reasons for this difference and the impact that it has on the utility of robot manipulators will be addressed in a later section of this chapter. In the following paragraphs, we present the results of several simple tests that are designed to illustrate the repeatability and accuracy of a PUMA 560 manipulator. The purpose behind this example is to illustrate the relative levels of repeatability and accuracy in a commonly available robot manipulator.

The first experiment is an investigation of repeatability. The manipulator, a PUMA 560, and the working environment are shown in Figure 1.2. The end effector used in these experiments consists of five tooling balls rigidly fixed and defined in a tool coordinate system. This end effector is illustrated in Figure 1.3. The device used to measure the location of the end effector is a small coordinate

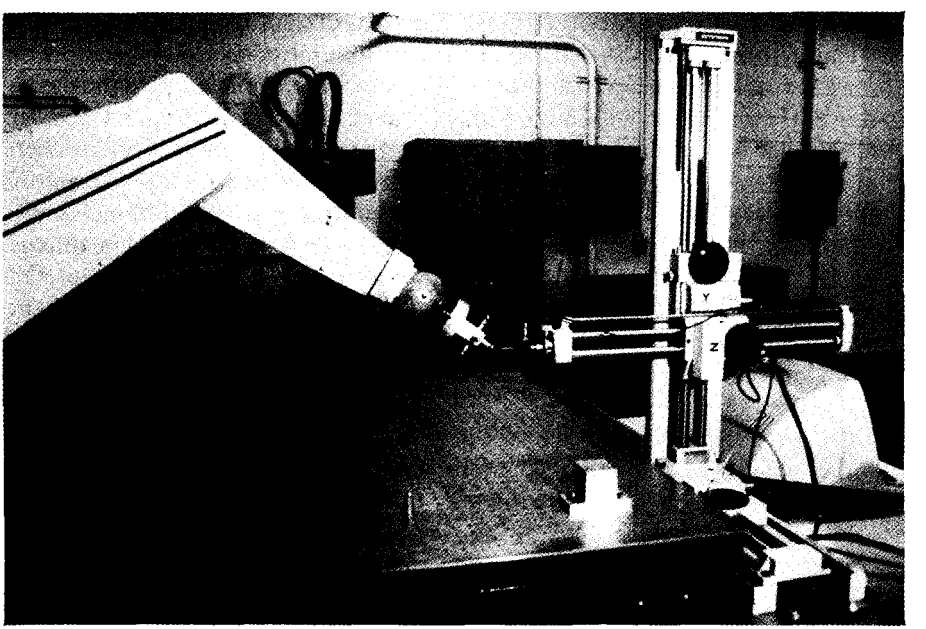


Figure 1.2. PUMA 560 and coordinate measuring machine.

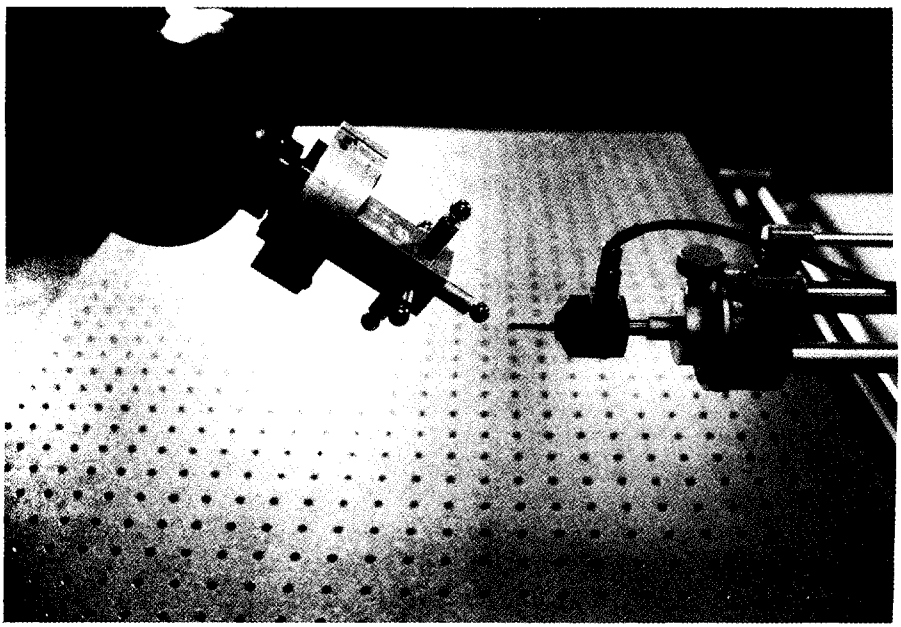
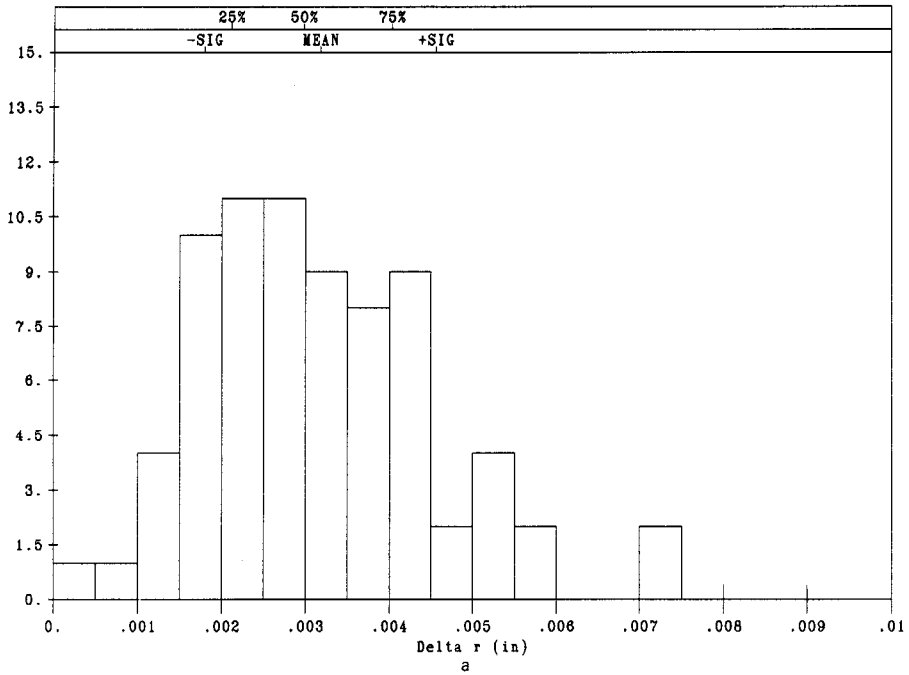


Figure 1.3. End effector for repeatability and accuracy experiments.

measuring machine (CMM). The CMM is equipped with a trigger probe that may be positioned arbitrarily in its working volume. When the trigger probe contacts an object, the Cartesian coordinates of the center of the trigger probe are recorded. To determine the pose of the manipulator end effector, the trigger probe is moved so that it contacts several points on the surface of one of the tooling balls in the end effector. A minimum of four points must be used to determine the center of the tooling ball. This process is then continued until the centers of three of the tooling balls have been located in the task space. These three points are then used to precisely determine the end effector pose. The end effector is equipped with five tooling balls to ensure that the CMM can reach at least three of the tooling balls for any orientation of the end effector.

The first experiment is intended to illustrate the repeatability of the manipulator. To accomplish this, the end effector is moved into an arbitrary pose and the joint displacements of the PUMA are recorded. The end effector pose is also measured with the CMM. The manipulator is then moved away from the initial pose and commanded to return to the taught configuration. After returning, the end effector pose is again measured with the CMM. The process of moving away and then returning to the initial pose is repeated 75 times. For each of the 75 poses, the distance, r, between the origin of the tool coordinate system in the current pose and the origin of the tool coordinate system in the initial pose is determined. The measure of variation in orientation is accomplished by recognizing that a rigid body may be changed from an arbitrary orientation to



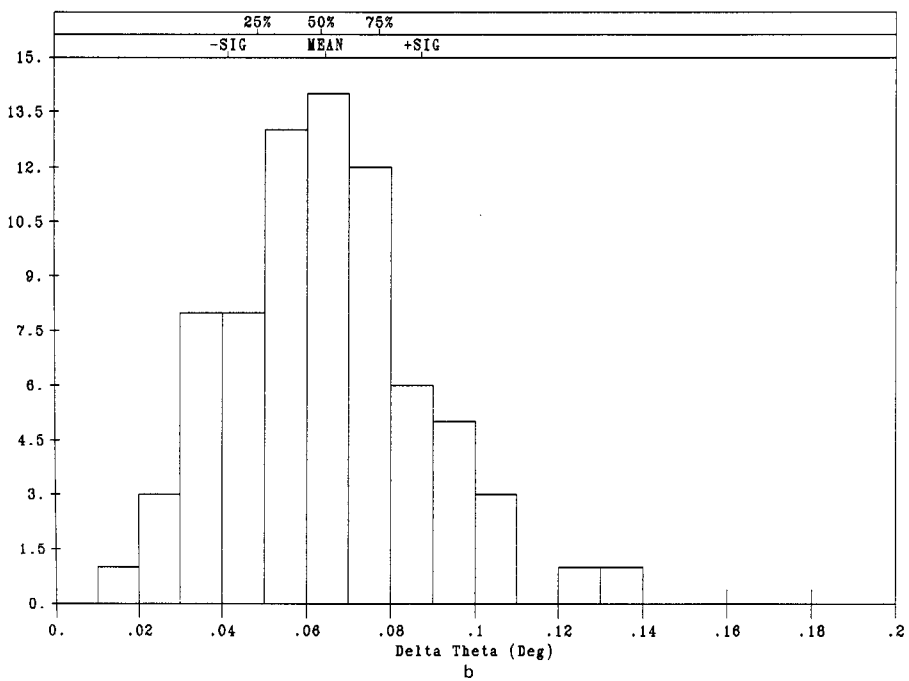
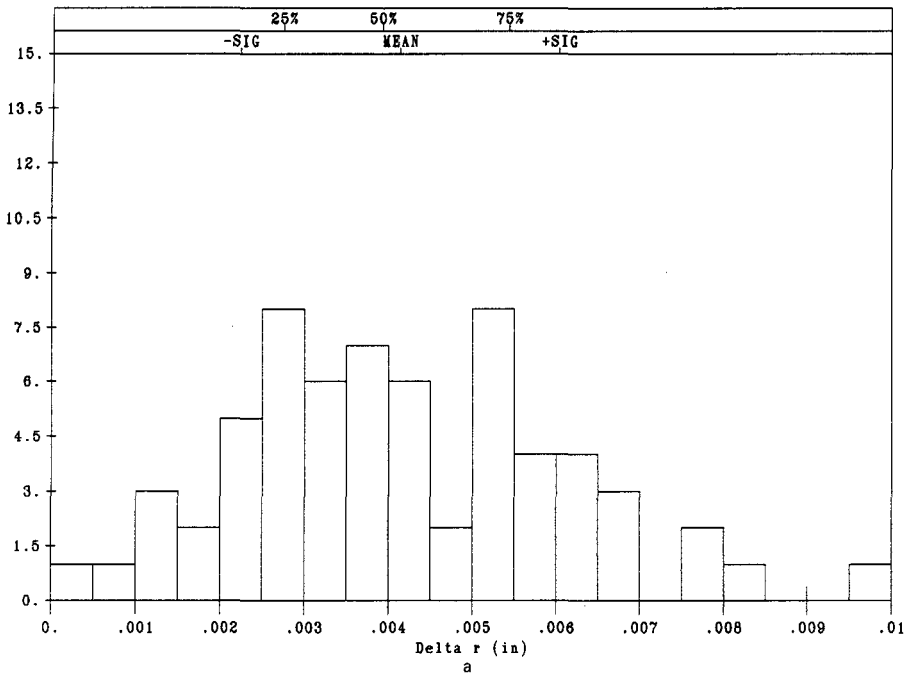


Figure 1.4. Repeatability results—Experiment A.

any other orientation with one single rotation about an axis in space. For each measured pose, the total rotation angle, θ, necessary to change from the current orientation to the initial orientation was determined. If the manipulator has perfect repeatability, the tool coordinate system will always return to the same pose. The variation in position and orientation that is measured in this experiment, therefore, tends to illustrate the levels of repeatability in the robot. The results of this experiment are illustrated in the histograms shown in Figure 1.4a and Figure 1.4b. These figures each consist of 20 bars that represent the number of times a measurement fell within the range indicated on the horizontal axis. For example, Figure 1.4a shows that only one pose had a position error, r, that fell between 0.0 and 0.0005 in. while eleven poses had a position error that fell between 0.0025 and 0.0030 in. The figures illustrate that both position and angular error are approximately normally distributed about a mean value. The position error, r, has a mean of 0.0032 in. and a standard deviation of 0.0014 in. The orientation error, θ, has a mean of 0.0642° with a standard deviation of 0.0230°. Although it has been shown [9] that repeatability can vary from one location in the working volume to another, it is usually assumed that a manipulator will have fairly constant repeatability over large parts of the working volume. This property can be illustrated by repeating the experiment described above for another manipulator pose. For the second pose, the manipulator end effector was moved to the opposite end of the CMM working volume. The results of the second experiment are illustrated in Figure 1.5a and 1.5b. In this pose, the position error has a mean of 0.0041 in. with a standard deviation of 0.0019 in. The angular error has a mean of 0.0905° with a standard deviation of 0.0323°. Although not exactly the same as in the previous experiment, these values indicate that the repeatability of the robot does not change significantly from one pose to the next. As mentioned earlier, the various aspects of repeatability will be discussed in more detail in Chapter 7. For the present discussion, it is important only to remember that the positioning repeatability of the manipulator is typically better than 0.007 in. and the orientation repeatability is usually better than 0.16°. Also, it is important to keep in mind that the repeatability does not vary significantly over large portions of the working volume.

With the results of the repeatability experiment in mind, we will now describe an experiment to illustrate the accuracy of the same manipulator. In this test, we will command the manipulator to move to nine specific poses in the working volume shared by the CMM and the robot. At each pose, the CMM is used to measure the actual pose of the robot which may then be compared with the commanded pose. The same measures of position error, r, and orientation error, θ, may be used again to quantify the deviation of each pose from the commanded pose. The results of this experiment are given in Table 1.1. Clearly there is a significant difference between the errors in accuracy and those for repeatability. A typical repeatability position error, r, was 0.005 in. while several of the accuracy position errors were in excess of 0.500 in. The difference in orientation errors is also significant, varying from an average of approximately 0.070° in the repeatability experiment to one error of almost 2° in the accuracy experiment. In



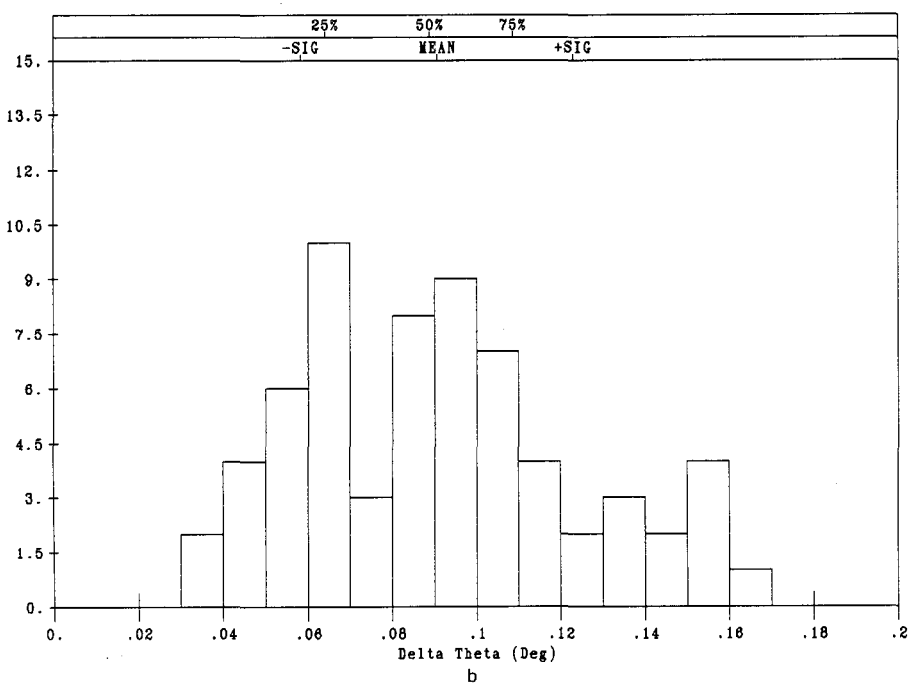


Figure 1.5. Repeatability results—Experiment B.

TABLE 1.1. Accuracy Experiment

Pose Number	Position Error (r) in.	Orientation Error (θ) deg
1	0.456	0.619
2	0.454	0.396
3	0.564	0.665
4	0.481	0.646
5	0.496	1.904
6	0.518	0.641
7	0.363	1.116
8	0.469	0.748
9	0.153	0.757

addition to the wide difference in magnitude of the errors, the variation of the robot accuracy across the workspace is also apparent. For example, we see a maximum position error of 0.564 in. and a minimum of only 0.153 in. The accuracy errors are so large that one would assume that there is an error in the location of the robot base coordinate system. If the actual location of the robot base is not precisely known in the world coordinate system, we would not expect accurate positioning of the end effector. To eliminate such errors, great care was taken in the experiment to precisely determine the robot base location for this experiment. In fact, the base location which minimized the accuracy errors for the nine poses was used. We may be confident, therefore, that the level of accuracy indicated in Table 1.1 is valid for the robot under study.

The experiments described above were designed to illustrate two aspects of precision that are common to most robot manipulators. First, the repeatability of the manipulator is quite good when compared with accuracy. In fact it is not uncommon to see accuracies that are several orders of magnitude worse than that of the manipulator repeatability. The second point to note is that manipulator repeatability is relatively constant across the work volume while the accuracy can vary significantly.

1.2 RELATIONSHIP BETWEEN MANIPULATOR PRECISION AND PROGRAMMING

In the previous section, we attempted to demonstrate that robot manipulators have significant differences in their level of repeatability and accuracy. We now want to discuss the impact that this difference has on the use of a robot in an industrial environment. A robot task may be programmed in one of two ways. The first, and most common approach to programming is to have an operator move the manipulator through the set of key configurations that make up the task. Each important configuration is stored in the robot controller and the task is accomplished by having the controller move the robot through the appropriate

series of stored configurations. During the programming process, the robot motion is controlled by an operator who uses a device called a *teach pendant.* With the teach pendant, the operator manipulates the various joints of the robot until the end effector is in the desired position and the robot configuration is stored in the controller. Clearly, this process can be quite time consuming and can require a skilled operator to produce a usable program. We will refer to this type of programming as the *teaching method.*

As mentioned above, the teaching method can be quite time consuming. Since lengthy manufacturing delays for robot programming are not desirable, it would be beneficial to develop the robot program away from the actual manufacturing line and simply transfer the completed program to the robot. This concept is referred to as *off-line programming.* Off-line programming may be accomplished with another robot in a laboratory environment or with a computer simulation of the robot and its working environment.

From the descriptions above, it would seem that off-line programming is so desirable that it would be by far the most widely used approach. This, however, is not the case. Only a few applications have been reported that utilize true off-line programming. The reason for this is explained by again considering the disparity between repeatability and accuracy. In the teaching method, a series of robot configurations that make up the task are stored. It is important to note that these configurations are defined by the joint displacements of the manipulator. In other words, the task is stored in joint space. During the teaching process, the operator ensures that the end effector is in the the appropriate pose at each task point. The conversion from task space to joint space is therefore made at each key task point with a visual confirmation by the operator. Since the key task points have been previously taught, the ability of the robot to attain these poses is measured by the manipulator repeatability. Off-line programming, on the other hand, relies on the assumption that a correct joint space description of a pose may be determined from the task space description. For example, assume that a robot has been set up in a laboratory and used to program a task. At each key task point, the end effector has been moved into position and the joint displacements have been recorded. It is now desired to move the program to another robot on the factory floor. Although the robots are repeatable, they are probably not accurate. This implies that for a given set of joint displacements, each robot will go to a significantly different pose. In other words, inaccurate manipulators have a different relationship between the joint space and task space descriptions of a pose for each robot and programs cannot be transferred from one machine to another.

To overcome problems associated with both repeatability and accuracy, some robot users have resorted to a third class of robot programming, which we will refer to as *workspace feedback.* For portions of the task that do not require high precision, either the teach method or off-line programming may be used. For those parts of the task that require high precision, the workspace is instrumented in a way that will provide the actual end effector pose to the controller. The controller may then make whatever correction is necessary to complete the task.

TABLE 1.2. Relationship between Programming Approach and Robot Precision

Type of Programming	Most Significant Aspect of Precision	Available Level of Precision
Teach method	Repeatability	Moderate to high
Workspace feedback	Resolution	Very high
Off-line	Accuracy	Low

For example, one typical manipulator application is the placement of surface mounted electronic components. Most manipulators accomplish this task with the aid of a vision system that provides information about the relationship between the current position of the component to be mounted and the desired placement. The digitized image provides the information necessary for the controller to correct errors resulting from either repeatability or accuracy.

It is clear from the discussion above that the various methods of programming are closely related to the different aspects of precision. These relationships have been summarized in Table 1.2. As illustrated in the table, each approach to programming is most dependent on a different aspect of robot precision. The teach method requires that the manipulator be as repeatable as possible. Although highly repeatable robots are available, the teach method of programming is time consuming and must be repeated each time a manipulator is replaced, moved, or its environment modified. The workspace feedback approach makes the manipulator part of a closed loop feedback system. When this is the case, resolution becomes the most significant aspect of precision. Since repeatability and accuracy errors are eliminated by the feedback device, it is necessary to ensure only that the robot has sufficient resolution to respond to the commands generated by the controller. The resolution of most robots is excellent and well within the limits necessary for this type of programming. There is, however, a significant disadvantage to this type of programming. When workspace feedback is employed, the pose sensor and its associated software are very task specific. At the present time, there is no general purpose sensing system that will precisely determine the pose of an end effector in a cluttered workspace for many different tasks. The workspace sensor system must be designed and programmed for a specific task. This process can be quite expensive and time consuming. In many applications, the design of the sensor system and programming of the feedback control through the controller can cost more than the manipulator itself. The other significant disadvantage of this approach is the loss of flexibility of the system. One of the attractive aspects of using robots is their flexibility and ease of reprogamming. Flexible manufacturing is supposed to offer the capability of quickly changing or modifying a task. When a complicated and task-specific sensor system has been added to a manipulator, this flexibility is lost. In fact, this desired flexibility is only a reality when off-line programming is used. New tasks may be planned and programmed without disturbing the operation of the current

task. When necessary, task changeover can be accomplished with a minimum of effort and impact on the production process. As indicated in Table 1.2, however, off-line programming relies on accurate robots that are not generally available.

1.3 MANIPULATOR CALIBRATION

As indicated in the previous discussion, the utility of robot manipulators would be significantly enhanced if they were made to be as accurate as they are repeatable. To enhance the accuracy, we must first understand the reason for the difference between accuracy and repeatability. As described above, repeatability is defined as the ability of the manipulator to return to a pose that has been stored in joint space. Accuracy, however, is the ability of the robot to move to a pose defined in task space. To achieve a pose that is defined in task space, the robot controller must convert the task space definition of the pose into joint space. The individual joints are then moved so that the desired configuration is obtained. The conversion from task space to joint space is accomplished by using a mathematical model of the manipulator. This mathematical model relates the joint displacements to the end effector pose and vice versa. When the mathematical model used by the robot controller to describe the robot motion differs from the actual geometry of the manipulator, the joint space definition of a pose defined in task space will not be accurate. To illustrate this concept, consider the simple 3 degree of freedom SCARA robot illustrated in Figure 1.6. In the design illustrated, all three axes of motion are intended to be parallel to each other and perpendicular to the base of the robot. For the purpose of this example, we will assume that when the joints have zero displacement, the arm is along the X axis

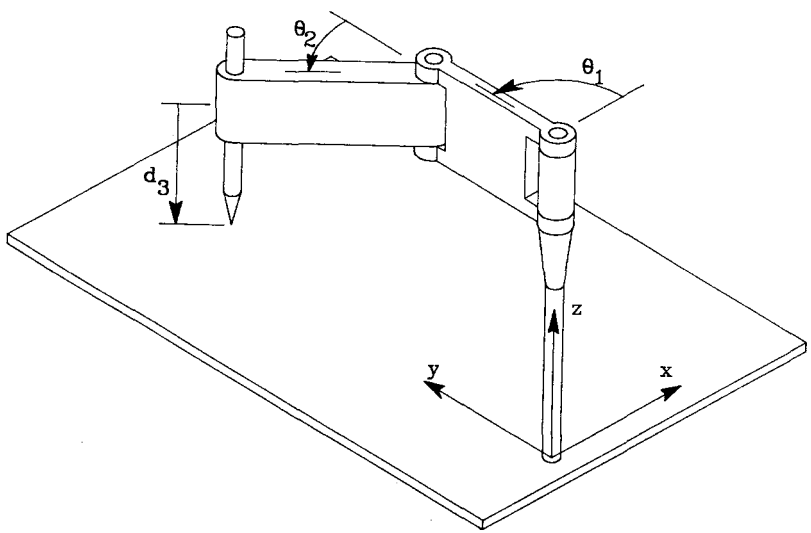


Figure 1.6. Three degree of freedom SCARA robot.

of the task space coordinate system and the tool point on the end effector has a displacement in the Z direction of 24 in. The relationship between the joint displacements, θ_1, θ_2, and d_3 and the position of the tool point, $\mathbf{r}_p$, is given by

$$r_{px} = 24\cos(\theta_1) + 24\cos(\theta_1 + \theta_2) \tag{1.3}$$

$$r_{py} = 24\sin(\theta_1) + 24\sin(\theta_1 + \theta_2) \tag{1.4}$$

$$r_{pz} = 24 - d_3 \tag{1.5}$$

In this example, Equations 1.3 through 1.5 represent the mathematical model that would be used to relate joint displacements to the position of the tool point. Now, assume that when the robot is constructed, a slight error is made in the alignment of axis 1 so that the axes of motion are not quite parallel with the base of the robot. The position of the tool point as predicted by our model will not be the position that is actually achieved by the manipulator. If the controller uses the model of the perfect robot to determine the joint displacements necessary to reach a pose defined in task space, an error will result. In this example, we will assume that the joint 1 axis is misaligned by 0.5° in the X–Z plane of the task space coordinate system. This would result in a positioning error that varies throughout the workspace and reaches a maximum value of over 0.083 in. when the arm is fully extended. When considering that a SCARA robot of this size could have a repeatability of better than 0.005 in., the significance of even slight variances between the mathematical model and the actual robot geometry becomes clear.

Since we have concluded that deviations between the mathematical model used in the controller and the actual arm geometry are a source of inaccuracy, it is clear that there are two basic ways of enhancing accuracy. The first would be to build every robot so that all of the various parameters match the "design" or "nominal" values as closely as possible. In other words, the manufacturing tolerances on every part would be extremely tight. Clearly, this approach is not feasible because of the excessive costs that would be involved. If we cannot make the robot match the model, then the second alternative is to make the mathematical model match the robot. This is the essence of manipulator calibration. Simply put, manipulator calibration is the process of defining an appropriate mathematical model and then determining the various model parameters that make the model match the robot as closely as possible.

1.3.1 Source and Significance of Manipulator Errors

Before beginning a more detailed discussion of manipulator calibration, it is beneficial to examine the most common sources of inaccuracy. Any difference between the actual robot geometry and that reflected in the mathematical model will produce some level of inaccuracy. Our purpose here is to examine a typical manipulator in an effort to identify the most probable sources of error and to make some assessment of their impact on the accuracy of the manipulator.

In a study of manipulator calibration, Whitney, Lozinski, and Rourke [14]

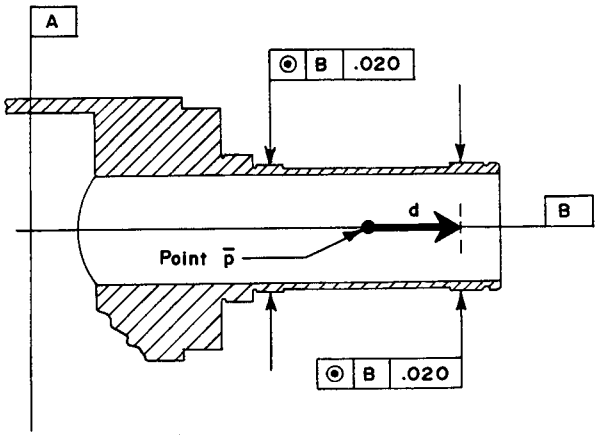


Figure 1.7. Revolute joint.

divided the sources of manipulator error into geometric and nongeometric errors. Geometric errors were defined as errors in the parameters that define the geometric relationships between the axes of motion. In other words, these are errors in those parameters that are not a function of the manipulator loading or motion. For example, errors in link length and joint axis orientation would be classified as geometric errors. Geometric errors usually arise during the construction of a manipulator and are a function of the tolerances used. To illustrate the relationship between assigned tolerances and variations in axis alignment, we will consider a brief example. Figure 1.7 is an illustration of a design for a robot component that will be the mounting surface for two bearings that make up a revolute joint. In the figure, a reference plane, A, and a reference line, B, are established. The bearing surfaces are required to be concentric to the line B to within 0.020 in. Since these surfaces will ultimately determine the location and orientation of the revolute axis, these tolerances may be used to examine the possible axis deviations. To illustrate this, we will define the point **p** to be located on the rotation axis halfway between the bearings. The unit vector **d** will lie along the rotation axis. If it is assumed that both bearings are at the tolerance limit in the vertical direction, point **p** will be displaced by 0.020 in. and **d** will remain parallel to its intended direction. If the left bearing is at the tolerance limit in the vertical direction and the right bearing is at the tolerance limit in the opposite direction, the point **p** will not move. The vector **d**, however, will make an angle, α, with the intended direction. In this example, α will be given by

$$\alpha = \tan^{-1}\left(\frac{0.040}{L}\right) \tag{1.6}$$

where L is the distance between the bearing centers. If we let L be 2 in., the angle α is 1.146°, which is an exceedingly large misalignment. This example illustrates that axis location can be relatively insensitive to assignment of tolerances. Axis

orientation, however, may be highly sensitive to bearing placement. This is especially true if the bearings are close together. In many instances, the tolerances necessary to ensure a precise axis alignment are unreasonably tight and would significantly increase manufacturing costs.

Other sources of error are termed nongeometric errors. Nongeometric errors require knowledge of the robot loading or motion. Joint compliance, gear backlash, dynamic errors in the controller, and bending of the links would be examples of nongeometric errors.

The source and significance of manipulator errors are topics that have attracted the attention of a number of researchers. There has been a good deal of controversy over which sources of error are the most significant and, therefore, the most important in the calibration process. At the present, it appears that there is no simple answer to this question. For example, Whitney, Lozinski, and Rourke [14] determined that the most significant error sources for the robot that they examined were several nongeometric errors. The robot used in their study was a PUMA 560. On the other hand, Judd and Knasinski [5] examined an Automatix AID 900 manipulator and found that the geometric errors were responsible for approximately 95% of the measured error and the nongeometric errors appeared to be almost negligible. Stone [13] reported a study of several PUMA 560 manipulators where it is shown that reduction of geometric errors caused a significant improvement in accuracy, but the effect of nongeometric errors was not studied.

At the present, it seems that the significance of various error sources depends highly on the particular robot or robot design that is being studied. For example, if a robot manufacturer develops a design that places high stress on the drive components we might expect to see significant nongeometric errors such as joint compliance and gear backlash. This would be especially true if tight tolerances were held during robot construction to ensure that the joint axes were positioned and oriented as designed. Conversely, if close attention is paid to the design of the controller and drive train for each joint, we would not expect to see the same level of significance in the nongeometric errors. If the design is such that the orientation of a joint axis is particularly sensitive to the location of a bearing, we might expect to see normal manufacturing tolerances lead to significant geometric errors. This train of thought seems to suggest that modifications in manipulator design would lead to enhanced accuracy. Although this is true, the tradeoffs that always exist in the design process will at times dictate that accuracy issues give way to more pressing concerns such as dexterity, weight, or cost. The point that we are trying to make is that the most significant error sources will be determined by the design and manufacturing process for a given robot model. This should be kept in mind when developing a calibration procedure for a given manipulator.

1.3.2 Levels of Manipulator Calibration

Since the significant sources of error can vary from one robot design to the next, calibration procedures can vary widely in their scope and complexity. For

example, some robot calibration procedures consider only the joint transducer information while others may involve changes in the kinematic or dynamic model of the robot. In an effort to classify most of the current approaches to robot calibration, we have chosen to define three levels of robot calibration.

For the purpose of this work, level 1 calibration shall be defined as "joint level" calibration. The goal is to determine the correct relationship between the signal produced by the joint displacement transducer and the actual joint displacement. This usually involves calibration of the kinematics of the drive and the joint sensor mechanisms. A level 2 calibration is defined as the entire robot kinematic model calibration. At this level the purpose of the calibration is to determine the basic kinematic geometry of the robot as well as the correct joint-angle relationships. Level 3 calibration is defined as "nonkinematic" (nongeometric) calibration. Nonkinematic errors in positioning of the end effector of a robot are due to effects such as joint compliance, friction, and clearance, as well as link compliance. Also, if the robot is under dynamic (rather than kinematic) control, correction for changes in the dynamic model of the robot constitutes a level 3 calibration.

1.3.3 What This Book Addresses

This text is intended to address issues related to level 1 and level 2 calibration. The decision to limit the discussion to these topics was made primarily because we feel that many significant issues involving level 1 and 2 calibration procedures have been resolved and that implementation of these techniques in an industrial environment is feasible. Also, the models that result from these calibration procedures are relatively easy to implement in existing robot controllers. Level 3 calibration, on the other hand, is still very much a research issue. The number of parameters involved at this level is significantly higher and, hence, the complexity of the associated data collection and identification procedures is much greater. Furthermore, to successfully implement the results of a level 3 calibration, the robot controller must make significant use of the manipulator dynamics. At the present, the vast majority of controllers for commercially available robots use a dynamic model only for joint level control. Joint coordination in these controllers is obtained from a kinematic model. A more detailed look at level 3 calibration may be obtained by reviewing the following papers [1–4, 6–8, 10–12].

1.4 THE CALIBRATION PROCESS

This section will provide an overview of the complete calibration process to which the remainder of the book is directed. In doing so, we will use terms that will be defined in detail in later chapters. The novice therefore may not fully understand the details of what follows but perhaps the strategy of calibration will be established and hence provide a framework for later chapters. This

strategy may best be explained by reconsidering the relationship between joint space and task space as expressed in Equation 1.2.

$$\mathbf{P} = f(\boldsymbol{\eta}, \boldsymbol{\theta}) \tag{1.7}$$

where $\mathbf{P}$ is the end effector pose as defined in task space, $\boldsymbol{\theta}$ is the vector of joint displacements, and $\boldsymbol{\eta}$ represents the set of constants used in the model. The goal of manipulator calibration has been defined as defining an appropriate functional form for Equation 1.7 and then determining the coefficients, $\boldsymbol{\eta}$, that make the model match the performance of the actual robot as closely as possible. This process falls conveniently into four sequential operations: modeling, measurement, identification, and implementation, each of which relates to the above equation. These areas will be dealt with in detail in subsequent chapters, however an overview of each is now given.

1.4.1 Modeling

The first step in the calibration process is the determination of a suitable functional form for Equation 1.7. Examination of the literature on kinematic modeling shows a wide variety of models used by different researchers in the field, and a number of questions arise as to how the models vary and which model should be used for a given manipulator. These questions are addressed through the concepts of completeness, equivalence, and proportionality and will be dealt with in detail in Chapter 2.

1.4.2 Measurement

The second step in the calibration process is measurement. The goal of the measurement process is to accurately determine either the end effector pose, or some subset of the pose, for a set of robot joint displacements. A typical measurement data set is obtained by moving the robot to some location, i, in the workspace, recording the joint displacements, $\boldsymbol{\theta}_i$, and then using an external measuring system to determine some portion of the pose, $\mathbf{P}_{mi}$. The robot is then moved to another location and the process repeated, continuing for as many measurements as necessary.

There are two aspects to the measurement process that need to be given careful consideration. The first is what measurement system should be used, and the second is how to plan the observation strategy correctly. There are only a few systems that have the necessary precision to make adequate pose or partial pose measurements. Each has its own characteristics such as precision, speed and ease of use, level of measurement noise, cost, and the amount of information obtained from each robot pose. These systems are discussed in detail in Chapter 3. In general, the measurement process is time consuming, laborious, and prone to human error. There is some benefit, therefore, in minimizing the number of

measurements that have to be taken, without compromising the end result of the calibration process. These issues will be addressed in Chapter 6, which presents a case study of the calibration of a PUMA 560 manipulator.

1.4.3 Identification

The identification process may be explained by considering the following equation

$$\delta \mathbf{P}_i = \mathbf{P}_{mi} - \mathbf{P}_{pi} \tag{1.8}$$

where $\mathbf{P}_{mi}$ is a measured pose and $\mathbf{P}_{pi}$ is the pose predicted by the model at the ith measurement location. Using Equation 1.7, this expression may be rewritten as

$$\delta \mathbf{P}_i = \mathbf{P}_{mi} - f(\boldsymbol{\eta}, \boldsymbol{\theta}_i) \tag{1.9}$$

where $\boldsymbol{\theta}_i$ is the set of joint displacements associated with measurement position i. The vector $\delta \mathbf{P}_i$ gives an indication of the difference between the pose predicted by the model and the measured pose for the given joint displacements. The purpose of the identification step is to choose the vector of model coefficients $\boldsymbol{\eta}$ that will minimize $\delta \mathbf{P}_i$ in some sense for the set of measured poses.

There are a number of well-known approaches to identification that will be described in detail in Chapter 4. Issues such as the relationship of the measurement noise to the accuracy of the resulting parameters and identification oriented observation strategy planning will also be addressed.

1.4.4 Implementation

Implementation means using calibration information to improve manipulator performance. So far, the calibration process should have given us an accurate kinematic model, with known parameters, that allows an accurate relationship between the joint variables and tool pose. The implementation phase is perhaps the least generalized aspect of the process since the details of the actual implementation tend to be somewhat machine and task specific. There are, however some generic concepts that provide a framework for this aspect of the problem.

Since the objective of the previous three phases, modeling, measurement, and identification, has been to determine the best model of the manipulator, conceptually the implementation phase simply involves the modification of the nominal model embedded in the robot controller. Clearly any manipulator controller that is able to accept an externally specified pose and convert it to a set of joint variables is performing an inverse solution on some model. However, the detailed architecture of a particular robot controller is not usually available to the user, making it difficult to implement calibration data in this way. In situations in which a controller does not contain a model of the manipulator, as in cases in which tool to joint transforms are not provided and the arm is only

able to be "taught" poses, implementation in the existing controller cannot be achieved.

Solving the above problems is usually accomplished by implementing the calibration data in an off-line preprocessor. Since virtually all manipulators are capable of being driven by the specification of a set of joint angles, this preprocessor performs the conversion of each world specified pose into the corresponding joint set through an inverse kinematic solution of the accurate manipulator model. The sets of joint variables are then sent to the existing joint controller and executed by the manipulator. This process is usually done once for a particular task, although there is no reason why it cannot be done in real time if the preprocessor may be used exclusively by a single manipulator.

Although this represents a simplistic overview of how calibration data may be utilized, Chapter 5 addresses some of the more subtle issues involved. For example, although a particular manipulator may have a model that has a known, closed form inverse, the model resulting from the calibration process may not. This may require the use of iterative numerical methods to derive the joint variable set from a specified pose. This and other related issues will be considered later in the text.

1.5 CONCLUSION

In this chapter, we have tried to demonstrate that the low levels of accuracy that typically exist in robot manipulators significantly impact their utility in a typical manufacturing setting. Furthermore, it has been shown that the source of this inaccuracy is the deviation between the actual structure of the robot and the mathematical model used in the controller. The purpose of calibration is to enhance manipulator accuracy by modifying the model so that it more closely matches a particular manipulator.

In the remainder of this book, the details of the calibration process will be described. Chapters 2 through 5 are dedicated to the four steps of the calibration process: modeling, measurement, identification, and implementation. Each of these chapters presents the various approaches to these steps and identifies research issues that are yet to be resolved. Chapter 6 details the calibration of a PUMA 560 robot. This case study is included to demonstrate the application of the calibration to an actual industrial manipulator. The final chapter overviews several approaches to measuring the performance of robot manipulators so that enhancements in accuracy may be properly quantified.

REFERENCES

[1] C. H. An, C. G. Atkeson, and J. M. Hollerbach. Estimation of inertial parameters of rigid body links of manipulators. In *Proceedings of 24th Conference on Decision and Control*, December 1985.

[2] B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the PUMA 500 arm. In *Proceedings IEEE Robotics and Automation Conference*, 1986.

[3] C. G. Atkeson, C. H. An, and J. M. Hollerbach. Estimation of inertial parameters of manipulator loads and links. *International Journal of Robotics Research*, 5(3), Fall 1986.

[4] C. G. Atkeson, C. H. An, and J. M. Hollerbach. Rigid body load identification for manipulators. In *Proceedings of 24th Conference on Decision and Control*, December 1985.

[5] Robert P. Judd and Al. B. Knasinski. A technique to calibrate industrial robots with experimental verification. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, pp. 351–357, April 1987.

[6] W. Khalil, M. Gautier, and J. F. Kleinfinger. Automatic generation of identification models of robots. *International Journal of Robotics and Automation*, 1(1), 1986.

[7] P. K. Khosla and T. Kanade. Parameter identification of robot dynamics. In *Proceedings of 24th Conference on Decision and Control, Ft. Lauderdale, FL*, December 1985.

[8] H. Mayeda, K. Osuke, and A. Kangawa. A new identification method for serial manipulator arms. In *Proceedings IFAC 9th World Congress, Budapest Hungary*, 1984.

[9] B. W. Mooring and T. J. Pack. Aspects of robot repeatability. *Robotica*, 5:223–230, 1987.

[10] A. Mukerjee and D. H. Ballard. Self-calibration in robot manipulators. In *Proceedings of 1985 IEEE International Conference on Robotics and Automation*, pp. 1050–1057, March 1985.

[11] H. B. Olsen and G. A. Bekey. Identification of parameters in models of robots with rotary joints. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.

[12] H. B. Olsen and G. A. Bekey. Identification of robot dynamics. In *Proceedings IEEE International Conference on Robotics and Automation*, April 1986.

[13] Henry W. Stone. *Kinematic Modeling, Identification, and Control of Robotic Manipulators*. Kluwer Academic Publishers, Boston, 1987.

[14] D. E. Whitney, C. A. Lozinski, and J. M. Rourke. Industrial robot forward calibration method and results. *Journal of Dynamic Systems, Measurement, and Control*, 108(1):1–8, March 1986.

CHAPTER 2

KINEMATIC MODELING FOR ROBOT CALIBRATION

The first step in any calibration procedure is to obtain a valid manipulator model. The purpose of the model is to relate the outputs of the joint displacement transducers to the pose of the end effector. There are two basic forms that any manipulator model may take. The *forward* or *direct* model computes the end effector pose given the joint transducer readings. The *inverse* model, on the other hand, determines the set of joint displacements that is necessary to achieve a specified pose. Although both models relate the same sets of information (joint displacements and pose), they are distinctly different in form and complexity. There are a number of methods of generating the forward model for a typical serial link manipulator. Most of these methods are easy to implement and lead to a unique relationship between the joint transducer displacements and the pose. The inverse model, however, can be quite difficult to derive and there is no easily applied methodology that will work for any robot geometry. The inverse model may also exhibit multiple solutions. This is reasonable since a given end effector pose may be obtained with several manipulator configurations.

Both the forward and inverse models come into play during the calibration process. Fortunately, the inverse model is used only during the implementation phase and will be addressed in Chapter 5. The model referred to in the "modeling phase" of calibration is the forward model. In this chapter, we will consider forward models for both Level 1 and Level 2 calibration procedures. In the discussion of Level 2 models, we will review the formalism established by Denavit and Hartenberg [5] and then demonstrate the extension of this method to relate variations in the kinematic parameters to variations in the pose. Readers that have not previously been introduced to kinematic modeling may wish to refer to one of the following books for a more detailed discussion of the fundamentals of kinematic modeling [1, 4, 10, 22].

2.1 LEVEL 1 MODELS

A Level 1 calibration has been defined as "joint level" calibration. The purpose is to correctly relate the signal from the joint displacement transducer to the actual joint displacement. There are a variety of joint displacement transducers and the more popular devices are described in Chapter 3. Since most manipulator joints are revolute or prismatic, these devices are usually rotary or linear and produce a digital signal or analog voltage. In some designs, the transducer is mounted directly to the joint axis. If this is the case, the model is simply the equation necessary to relate the transducer signal to the joint displacement. In the case that the position transducers are on a prime mover such as a motor shaft, the model includes the kinematics of the drive system. If we define the signal from the transducer as η_i and the actual joint displacement as θ_i, the following relationship holds:

$$\theta_i = h_i(\eta_i, \gamma_i) \tag{2.1}$$

where $h_i(\ )$ is the appropriate input–output functional relationship in explicit form and the vector γ_i represents the vector of parameters in the function $h(\ )$. In the large majority of cases the function $h(\ )$ is assumed to be linear and can be written as

$$\theta_i = k_{i1}\eta_i + k_{i2} \tag{2.2}$$

In this model the vector γ_i will be $[k_{i1}, k_{i2}]^T$. The purpose of the Level 1 calibration, therefore, would be to determine the values of the vector γ_i correctly.

As an example of this type of model, we will consider a situation in which we have an incremental encoder connected directly to a revolute axis. The output from the encoder is a pulse count, n. In this case, the joint angle, θ, will be given by

$$\theta = k_1 n + k_2 \tag{2.3}$$

where k_1 is the angle represented by each pulse and k_2 is the joint angle when the pulse count is zero. The purpose of a calibration would be to determine the appropriate values of k_1 and k_2. In this example, the value of k_1 would be determined by simply obtaining the number of lines on the encoder and checking to see if the electronic counter in the controller multiplies the count by 1, 2, or 4. If, for example, we have a device with 500 lines and a count multiplier of 2, there would be 1000 counts per revolution. If the device is connected directly to the joint shaft, this gives a value of 360°/1000 counts or $k_1 = 0.36$ deg/count. Given the physical arrangement described in the example, there would be no variation in this value and, therefore, it could be treated as a constant and not included in the calibration process. The value of k_2, however, gives the joint angle when the pulse count is zero. This value must be set every time the controller is powered up since the encoder gives a relative pulse count and not an absolute

reading of joint angle. The value, k_2, is usually referred to as the joint offset and may be determined by moving the joint to a known angle and resetting the pulse counter. Many robots automatically do this as a part of the start-up sequence and the process is referred to as the homing or initialization procedure. In actuality this is an automated Level 1 calibration using the model described above.

In cases involving high precision it may become necessary to develop a more sophisticated model to describe the relationship between the transducer signal and the joint displacement. This is usually the case if it is necessary to include drive train kinematics in the relationship. As an example, consider a revolute joint driven by a dc motor through a gear train. If the joint transducer is an encoder mounted on the motor shaft rather than the joint axis, nonlinearities in the gear train between the motor and the joint axis will affect the relationship between the joint angle and the encoder. If we assume that one of the gears is slightly eccentric, the relationship between joint angle and motor shaft angle will have a harmonic component. A valid model in this case might be

$$\theta = k_1 n + k_2 + k_3 \sin(k_4 n + k_5) \tag{2.4}$$

where n is the encoder count, k_1 and k_2 are the slope and joint offset, k_4 reflects the ratio to the eccentric gear, and k_5 is a phase angle. Whereas k_1 and k_4 may be treated as constants that are determined from the encoder specifications and joint design, k_2, k_3, and k_5 must be determined by calibrating the joint.

As illustrated above, the specific form of a Level 1 model is highly dependent on the type of nonlinearities that may exist in the joint. It is common to find nonlinearities from gear backlash and component compliance. Some joint transducers also produce nonlinearities that may be included in the calibration model.

2.2 LEVEL 2 MODELS

Once the Level 1 model has been determined, a model must be developed that relates the joint angles to the end effector position. A number of different approaches exist for developing the kinematic model of a robot manipulator. The most popular method has been the procedure established by Denavit and Hartenberg [5], which is based on homogeneous transformation matrices. This procedure consists of establishing coordinate systems on each joint axis. Each coordinate system is then related to the next through a specific set of coefficients in the homogeneous transformation matrices. This modeling procedure will be reviewed to acquaint the reader with the nomenclature to be used throughout the rest of the text and is not intended to be a complete introduction to kinematic modeling. Those who are unfamiliar with kinematic modeling may wish to refer to one of the the following books for a more detailed discussion of kinematic modeling [1, 4, 10, 22].

Once the Denavit–Hartenberg (DH) procedure has been reviewed, the model will be used to demonstrate the derivation of the relationship between variations

in the model parameters and the predicted end effector pose. This relationship will first be applied to a single link and then to an entire manipulator. This analysis is followed by a discussion in the limitations of the DH that are significant to calibration and a review of models that overcome these limitations.

Before beginning a more detailed discussion of modeling, it is appropriate to consider the notation that will be used in the following sections. Since the mathematical expressions will involve scalars, vectors, and matrices, it is important to establish a consistent notation to enhance clarity. A number of different notations have been proposed for kinematic modeling and the choice is somewhat a matter of personal taste. We have chosen to follow the notation used by Paul [22] since it has been widely adopted. All scalars will be represented by an uppercase or lowercase character that is not shown as boldface. For example, θ, η, k_1, and k_2 would represent scalar values. Vectors will be denoted by boldface, lowercase characters such as $\mathbf{r}$ and $\boldsymbol{\gamma}$. Matrices will typically be used to represent coordinate frames or transformations and will be denoted by boldface, uppercase characters. Examples of matrices would be $\mathbf{T}$ or $\mathbf{A}$. In many cases, subscripts will be used to designate various components or coordinate frames. In each case, the meaning of the subscript notation will be defined.

2.3 DENAVIT–HARTENBERG METHOD

One of the most fundamental problems in describing a working environment in which one or more robots operate, together with supporting equipment, is how to explain the relative positions of the various pieces of equipment. This is important since many robot operations are position driven. For example, a robot has to pick up a part from a certain location, put it down in another location, change end effectors by collecting a different gripper from yet another location, and so on. The study of kinematics reveals that a method exists allowing us to define these positions in a consistent and unambiguous manner. The method consists of attaching coordinate frames (or just frames) to each object or location of interest so that when the object moves, so does the frame. The problem then is reduced to one specifying the relationship between the frames. Fortunately, kinematics helps here too, since homogeneous transformations allow us to do just that. These concepts will be fully explained in the next paragraph; however, they are illustrated in Figure 2.1, which shows a robot workcell. Each object of interest together with some important locations used by the robot, as well as with their coordinate frames, are shown.

In the analysis of kinematic structures consisting of serial links and joints, accepted methods for defining the position and orientation of one link with respect to another are in common use. We have seen in the previous paragraph that one method is to allocate kinematic frames to each of the robot links and then to define the position of the robot by specifying the transformation from

one link to the other. In this way, the spatial orientation of quite complex kinematic structures may be specified in a unified, straightforward method. The DH method involves the allocation of coordinate frames to each link using a set of rules to locate the origin of the frame and the orientation of the axes. The position of consecutive links is then defined by the homogeneous transformation matrix, which transforms the frame attached to link $n-1$ into the frame fixed to link n. This transformation is obtained from simpler transformations representing the three basic translations along, and three rotations about, the frame's x, y, and z axes. These fundamental transforms, expressed in a 4×4 matrix notation, may be shown as follows:

$$\mathbf{T}(x', y', z') = \begin{bmatrix} 1 & 0 & 0 & x' \\ 0 & 1 & 0 & y' \\ 0 & 0 & 1 & z' \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.5}$$

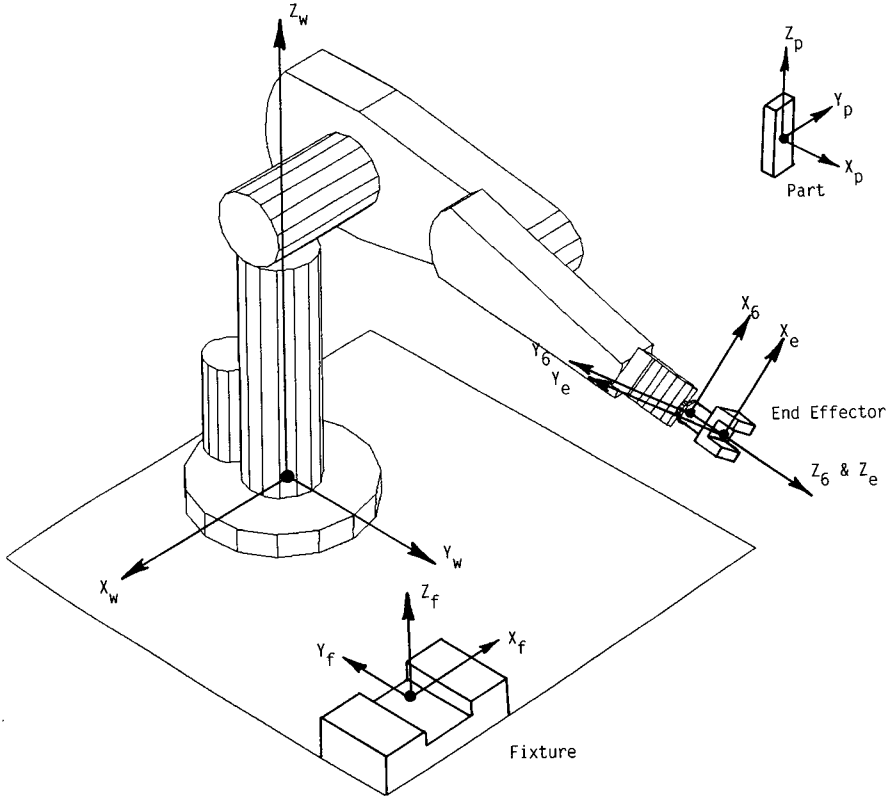


Figure 2.1. Components of a robotic workcell.

$$\mathbf{R}(x,\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$\mathbf{R}(y,\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$\mathbf{R}(z,\theta_z) = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

where $\mathbf{T}(x',y',z')$ implies a translation given by the vector $\mathbf{r} = [x',y',z']^T$ and $\mathbf{R}(x,\theta_x)$ implies a rotation of θ_x about the x coordinate axis.

When applied to the linkage shown in Figure 2.2, for example, we can use Equations 2.5–2.8 to define the position of link 2 with respect to link 1 by specifying the transformation, commonly called the **A** matrix, which transforms frame 1 into frame 2. It may be seen that this transformation takes the following form:

$$\mathbf{A} = \mathbf{T}(d_1,0,0)\mathbf{R}(z,\theta_1)\mathbf{T}(d_2,0,0) \tag{2.9}$$

or

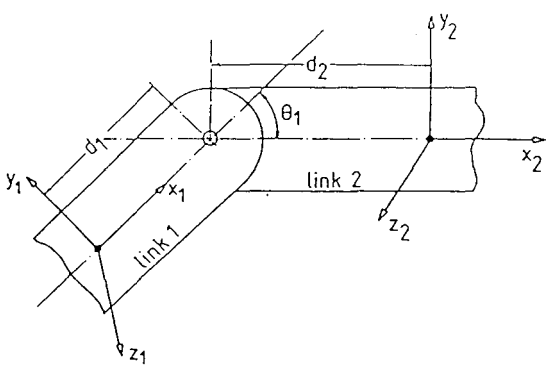


Figure 2.2. Two link mechanism.

$$\mathbf{A} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & d_2 + d_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & d_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

For more general linkage structures, which are usually three dimensional in nature, the transformations are more complex. In such cases, much labor may be avoided by establishing a common methodology for allocating the frames on the links, and defining a common set of transformations to get from one link to the next. In his description of the DH method, Paul [22] derives the standard transformation for the noncoplanar kinematic chain shown in Figure 2.3. The process is begun by identifying the axis of motion for each joint. Next, the common normal between consecutive joint axes is then identified. The origin of coordinate frame n is then located at the intersection of joint axis $n + 1$ and the common normal between axis $n + 1$ and axis n. The z axis of coordinate system n points along the axis of joint $n + 1$ and the x axis is aligned with the common normal as shown in Figure 2.3. Assigning the frames in this manner allows the transformation matrix from frame $n - 1$ to frame n to be expressed as

$$\mathbf{A}_n = \mathbf{R}(z, \theta_n)\mathbf{T}(0, 0, r_n)\mathbf{T}(l_n, 0, 0)\mathbf{R}(x, \alpha_n) \tag{2.11}$$

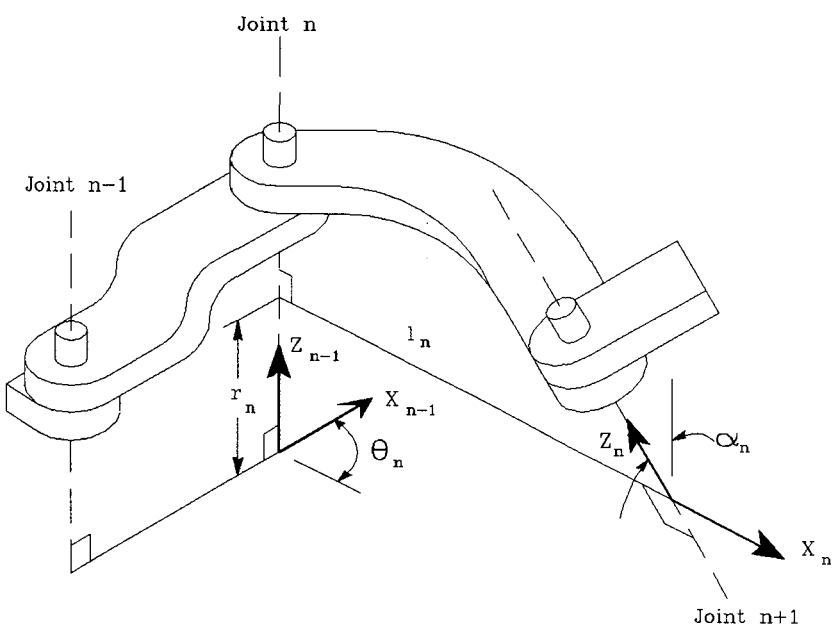


Figure 2.3. Kinematic frame allocation.

The above equation may be interpreted as a means of transforming frame $n-1$ to frame n by the following sequential steps

- Rotate frame $n-1$ about z_{n-1} by an angle θ_n, the joint angle
- Translate along z_{n-1} a distance r_n, the offset
- Translate along the rotated x_{n-1}, a distance l_n, the link length, and
- Rotate about x_n the twist angle α_n.

If the assignment of frames is adopted for all links, Equation 2.11 may be used as a recursive transformation relating the position of one frame with respect to the previous one. Just as in Equation 2.10, the transformation matrix is a function of the link geometry such as the length l_n, the twist α_n, the offset between the common normals r_n, and also the joint angle θ_n, This leads to the general form of the homogeneous transformation as follows:

$$\mathbf{A} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & l_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & l_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & r_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

For a revolute joint, the parameters l_n, α_n, and r_n are constants that describe the robot geometry and θ_n is the variable that describes the joint displacement. In a prismatic joint, only the orientation of the joint axis is important. The location of the origin of the coordinate system, therefore, is determined by moving the axis of the prismatic joint so that it intersects the axis of the next joint. This forces the length of the common normal, l_n, to be zero. In a prismatic axis, therefore, α_n and θ_n define the link geometry and r_n is the joint variable.

Since the specification of position and orientation of a rigid body in space requires six generalized coordinates, a robot manipulator will require six degrees of freedom to achieve complete dexterity within its workspace. A manipulator with six links and six joints will be described by six **A** matrices as defined by Equation 2.8, with all of the l, α, and r variables defined, and the instantaneous position of the arm defined by the six joint variables. As an example, we will consider the PUMA 560 arm, the analysis of which is well documented by Paul [22] and others. Figure 2.4 shows a schematic of this arm with each link, joint, and coordinate frame defined as shown. The base frame, 0, is fixed while the final frame, 6, defines the end of the manipulator. It is easily seen that the compound transformation

$$\mathbf{T}_2 = \mathbf{A}_1\mathbf{A}_2 \tag{2.13}$$

defines the transformation from frame 0 to frame 2 while

$$\mathbf{T}_3 = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3 \tag{2.14}$$

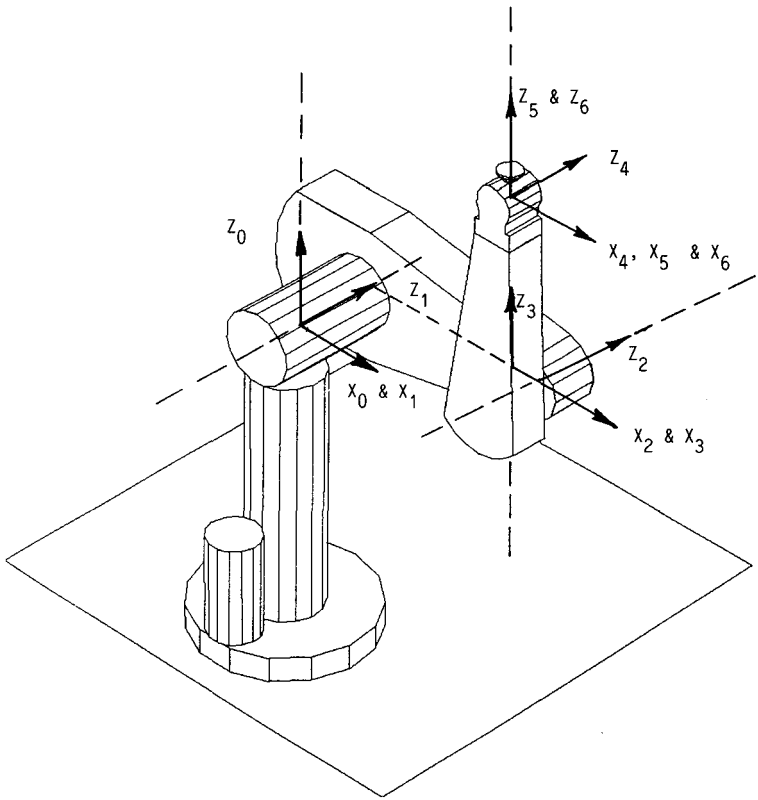


Figure 2.4. PUMA manipulator.

defines the transformation from frame 0 to frame 3, and so on. Continuing, we get

$$\mathbf{T}_6 = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4\mathbf{A}_5\mathbf{A}_6 \tag{2.15}$$

where $\mathbf{T}_6$ is a function of the six joint variables, and is a 4×4 homogeneous transformation describing the position and orientation of frame 6, attached to the end link of the manipulator, with respect to the base.

Equation 2.15 represents the forward model for a 6 axis manipulator. Since we will consider using this model for calibration, there are several aspects of this model that should be reviewed. If the DH procedure is followed, each revolute joint will require three constants to describe the link geometry and one variable to define the joint rotation. Each prismatic joint requires two constants for link geometry and one joint variable. In a revolute joint, the joint variable, θ_n, is measured from the x_{n-1} axis. A change in axis orientation, therefore, would change the reference position for the joint displacement. This implies that the robot "zero" or reference position is a function of the axis geometry. For example, if we have two manipulators with slightly different geometries, they will have two

different reference positions. If we command each robot to go to the configuration where all axis displacements are 0, each robot end effector will be in a different pose. Since the goal of calibration is to make two slightly dissimilar manipulators perform the same, each should have the same reference position. To meet the goals of calibration, therefore, some extension to the DH formalism to allow adjustment of the reference position is required. Another property of the DH model is that the location of each coordinate system is defined by the robot geometry. For example, the base coordinate system is constrained to lie on the axis of joint 1. In many robot geometries, this is inside the physical structure of the robot. Since the location of the robot in the workspace must be known, some means of accurately locating the base coordinate system must be available. If there are no physical references on the robot base and the base coordinate system is located inside the robot structure, location of the robot in the workspace can be difficult. Again, an extension to the DH formalism can address this problem. These and other limitations of the DH model are addressed further in Section 2.6. In paragraphs following Section 2.6, extensions to the DH model that address these problems will be discussed.

Before considering alternate models, however, we wish to examine the effect of variations in the kinematic parameters on the end effector pose. Although the specific mathematics required to accomplish this will depend on the model used, a general approach may be employed. This procedure is illustrated in the following section.

2.4 LINK KINEMATIC ERROR MODEL

The material presented in this section indicates how small errors about the nominal manipulator kinematics produce end point position and orientation changes, and is developed into a form that allows a calibration methodology to be formulated. Many approaches to the problem have been proposed [2, 3, 12, 13, 19], but the one given here follows that of Veitschegger and Wu [32]. We will use a standard DH model to illustrate the approach. Modification of this procedure for different models is easily accomplished.

Beginning with Equation 2.12 and assuming that all variations about the nominal kinematics may be accounted for by variations in θ_n, α_n, r_n, and l_n we have the change in a single transformation matrix $\mathbf{A}_n$ to be

$$d\mathbf{A}_n = \frac{\partial \mathbf{A}_n}{\partial \theta_n}\Delta\theta_n + \frac{\partial \mathbf{A}_n}{\partial \alpha_n}\Delta\alpha_n + \frac{\partial \mathbf{A}_n}{\partial \mathbf{r}_n}\Delta\mathbf{r}_n + \frac{\partial \mathbf{A}_n}{\partial l_n}\Delta l_n \tag{2.16}$$

by differentiating Equation 2.12 we get

$$\frac{\partial \mathbf{A}_n}{\partial \theta_n} = \begin{bmatrix} -s\theta_n & -c\theta_n c\alpha_n & c\theta_n s\alpha_n & -l_n s\theta_n \\ c\theta_n & -s\theta_n c\alpha_n & s\theta_n s\alpha_n & l_n c\theta_n \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \mathbf{Q}'_\theta \mathbf{A}_n \tag{2.17}$$

$$\mathbf{Q}'_\theta = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.18}$$

By representing

$$\frac{\partial \mathbf{A}_n}{\partial \theta_n} = \mathbf{A}_n \mathbf{Q}_\theta \tag{2.19}$$

$\mathbf{Q}_\theta$ can be written as

$$\mathbf{Q}_\theta = \mathbf{A}_n^{-1} \mathbf{Q}'_\theta \mathbf{A}_n = \begin{bmatrix} 0 & -c\alpha_n & s\alpha_n & 0 \\ c\alpha_n & 0 & 0 & l_n c\alpha_n \\ -s\alpha_n & 0 & 0 & -l_n s\alpha_n \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.20}$$

Similarly,

$$\frac{\partial \mathbf{A}_n}{\partial r_n} = \mathbf{A}_n \mathbf{Q}_r \tag{2.21}$$

$$\frac{\partial \mathbf{A}_n}{\partial l_n} = \mathbf{A}_n \mathbf{Q}_l \tag{2.22}$$

$$\frac{\partial \mathbf{A}_n}{\partial \alpha_n} = \mathbf{A}_n \mathbf{Q}_\alpha \tag{2.23}$$

where

$$\mathbf{Q}_r = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & s\alpha_n \\ 0 & 0 & 0 & c\alpha_n \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.24}$$

$$\mathbf{Q}_l = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.25}$$

and

$$\mathbf{Q}_\alpha = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.26}$$

Substituting these results back into Equation 2.16 yields

$$d\mathbf{A}_n = \mathbf{A}_n(\mathbf{Q}_\theta \Delta\theta_n + \mathbf{Q}_r \Delta r_n + \mathbf{Q}_l \Delta l_n + \mathbf{Q}_\alpha \Delta\alpha_n) \tag{2.27}$$

Following Paul [22] by defining an error matrix $d\mathbf{A}_n$ such that

$$d\mathbf{A}_n = \mathbf{A}_n \delta\mathbf{A}_n \tag{2.28}$$

we have

$$\delta\mathbf{A}_n = \mathbf{Q}_\theta \Delta\theta_n + \mathbf{Q}_r \Delta r_n + \mathbf{Q}_l \Delta l_n + \mathbf{Q}_\alpha \Delta\alpha_n \tag{2.29}$$

where from Equations 2.20–2.26

$$\delta\mathbf{A}_n = \begin{bmatrix} 0 & -c\alpha_n \Delta\theta_n & s\alpha_n \Delta\theta_n & \Delta l_n \\ c\alpha_n \Delta\theta_n & 0 & -\Delta\alpha_n & l_n c\alpha_n \Delta\theta_n + s\alpha_n \Delta r_n \\ -s\alpha_n \Delta\theta_n & \Delta\alpha_n & 0 & -l_n s\alpha_n \Delta\theta_n + c\alpha_n \Delta r_n \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.30}$$

This homogeneous transformation may be partitioned into a 3×1 displacement component $\mathbf{d}_n$ and a 3×1 rotational component $\boldsymbol{\delta}_n$ as follows:

$$\begin{aligned} \mathbf{d}_n &= \begin{bmatrix} \Delta l_n \\ l_n c\alpha_n \Delta\theta_n + s\alpha_n \Delta r_n \\ -l_n s\alpha_n \Delta\theta_n + c\alpha_n \Delta r_n \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ l_n c\alpha_n \\ -l_n s\alpha_n \end{bmatrix} \Delta\theta_n + \begin{bmatrix} 0 \\ s\alpha_n \\ c\alpha_n \end{bmatrix} \Delta r_n + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Delta l_n \end{aligned} \tag{2.31}$$

and

$$\begin{aligned} \boldsymbol{\delta}_n &= \begin{bmatrix} \Delta\alpha_n \\ s\alpha_n \Delta\theta_n \\ c\alpha_n \Delta\theta_n \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ s\alpha_n \\ c\alpha_n \end{bmatrix} \Delta\theta_n + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Delta\alpha_n \end{aligned} \tag{2.32}$$

By defining the following three vectors

$$\mathbf{k}_n^1 = [0 \quad l_n c\alpha_n \quad -l_n s\alpha_n]^T \tag{2.33}$$

$$\mathbf{k}_n^2 = [0 \quad s\alpha_n \quad c\alpha_n]^T \tag{2.34}$$

$$\mathbf{k}_n^3 = [1 \quad 0 \quad 0]^T \tag{2.35}$$

the translational and rotational errors at $\mathbf{A}_n$ due to the link parameter errors can be expressed in the following linear form

$$\mathbf{d}_n = \mathbf{k}_n^1 \Delta\theta_n + \mathbf{k}_n^2 \Delta r_n + \mathbf{k}_n^3 \Delta l_n \tag{2.36}$$

$$\boldsymbol{\delta}_n = \mathbf{k}_n^2 \Delta\theta_n + \mathbf{k}_n^3 \Delta\alpha_n \tag{2.37}$$

2.5 MANIPULATOR KINEMATIC ERROR MODEL

If we apply the analysis in the preceding section to an N link manipulator, we are able to derive the position and orientation errors at the tool frame due to the four link kinematic errors for each link of the arm. It is to be expected therefore that the manipulator kinematic error model will be comprised of $4N$ unknown link error parameters. We can express the deviation from the expected end position $\mathbf{T}_N$ by an error matrix $d\mathbf{T}_N$ where

$$\begin{aligned} \mathbf{T}_N + d\mathbf{T}_N &= (\mathbf{A}_1 + d\mathbf{A}_1)(\mathbf{A}_2 + d\mathbf{A}_2)\cdots(\mathbf{A}_N + d\mathbf{A}_N) \\ &= \prod_{n=1}^{N} (\mathbf{A}_n + d\mathbf{A}_n) \end{aligned} \tag{2.38}$$

Expanding Equation 2.38 and ignoring second-order products we get after some manipulation

$$\mathbf{T}_N + d\mathbf{T}_N = \mathbf{T}_N + \sum_{n=1}^{N} (\mathbf{A}_1 \ldots \mathbf{A}_{n-1} d\mathbf{A}_n \mathbf{A}_{n+1} \ldots \mathbf{A}_N) \tag{2.39}$$

Substituting 2.28 into 2.39 gives

$$\begin{aligned} d\mathbf{T}_N &= \sum_{n=1}^{N} (\mathbf{A}_1 \ldots \mathbf{A}_n) \delta\mathbf{A}_n (\mathbf{A}_{n+1} \ldots \mathbf{A}_N) \\ &= \sum_{n=1}^{N} \mathbf{T}_N (\mathbf{A}_{n+1} \ldots \mathbf{A}_N)^{-1} \delta\mathbf{A}_n (\mathbf{A}_{n+1} \ldots \mathbf{A}_N) \end{aligned} \tag{2.40}$$

defining the matrix $\mathbf{U}_n$ to the product of the $\mathbf{A}$ matrices from n to the end of the manipulator

$$\mathbf{U}_n = \prod_{i=n}^{N} \mathbf{A}_i \tag{2.41}$$

we have

$$d\mathbf{T}_N = \mathbf{T}_N \left[\sum_{n=1}^{N} \mathbf{U}_{n+1}^{-1} \delta \mathbf{A}_n \mathbf{U}_{n+1} \right] \tag{2.42}$$

Since we may write

$$d\mathbf{T}_N = \mathbf{T}_N \delta \mathbf{T}_N \tag{2.43}$$

we obtain

$$\delta \mathbf{T}_N = \sum_{n=1}^{N} \mathbf{U}_{n+1}^{-1} \delta \mathbf{A}_n \mathbf{U}_{n+1} \tag{2.44}$$

Following a different approach, which states that the error in $\mathbf{T}_N$ may be defined as small displacements from $\mathbf{T}_N$ of dx_N, dy_N, and dz_N, and small rotations δx_N, δy_N, δz_N about the x_N, y_N, and z_N axes respectively, from

$$\delta \mathbf{T}_N = \mathbf{T}(dx_N, dy_N, dz_N)\mathbf{R}(x, \delta x_N)\mathbf{R}(y, \delta y_N)\mathbf{R}(z, \delta z_N) \tag{2.45}$$

It may be shown that

$$\delta \mathbf{T}_N = \begin{bmatrix} 0 & -\delta z_N & \delta y_N & dx_N \\ \delta z_N & 0 & -\delta x_N & dy_N \\ -\delta y_N & \delta x_N & 0 & dz_N \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.46}$$

Paul [22] has shown that for the general matrix expression $\mathbf{T}^{-1}\Delta\mathbf{T}$ we may write the corresponding matrix as

$$\begin{bmatrix} 0 & -\boldsymbol{\delta}\cdot(\mathbf{n}\times\mathbf{o}) & \boldsymbol{\delta}\cdot(\mathbf{a}\times\mathbf{n}) & \boldsymbol{\delta}\cdot(\mathbf{p}\times\mathbf{n})+\mathbf{d}\cdot\mathbf{n} \\ \boldsymbol{\delta}\cdot(\mathbf{n}\times\mathbf{o}) & 0 & -\boldsymbol{\delta}\cdot(\mathbf{o}\times\mathbf{a}) & \boldsymbol{\delta}\cdot(\mathbf{p}\times\mathbf{o})+\mathbf{d}\cdot\mathbf{o} \\ -\boldsymbol{\delta}\cdot(\mathbf{a}\times\mathbf{n}) & \boldsymbol{\delta}\cdot(\mathbf{o}\times\mathbf{a}) & 0 & \boldsymbol{\delta}\cdot(\mathbf{p}\times\mathbf{a})+\mathbf{d}\cdot\mathbf{a} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.47}$$

where $\mathbf{T}$ may be generalized as a combination of four unit vectors, $\mathbf{n}$, $\mathbf{o}$, $\mathbf{a}$, and $\mathbf{p}$ as shown below.

$$\mathbf{T} = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.48}$$

The vector Δ is defined as

$$\Delta = \begin{bmatrix} \mathbf{d} \\ \boldsymbol{\delta} \end{bmatrix} \tag{2.49}$$

From 2.44, 2.46, and utilizing 2.47 we obtain

$$
\begin{aligned}
dx_N &= \sum_{n=1}^{N} [\mathbf{n}_{n+1}^u \cdot \mathbf{d}_n + (\mathbf{p}_{n+1}^u \times \mathbf{n}_{n+1}^u) \cdot \boldsymbol{\delta}_n] \\
dy_N &= \sum_{n=1}^{N} [\mathbf{o}_{n+1}^u \cdot \mathbf{d}_n + (\mathbf{p}_{n+1}^u \times \mathbf{o}_{n+1}^u) \cdot \boldsymbol{\delta}_n] \\
dz_N &= \sum_{n=1}^{N} [\mathbf{a}_{n+1}^u \cdot \mathbf{d}_n + (\mathbf{p}_{n+1}^u \times \mathbf{a}_{n+1}^u) \cdot \boldsymbol{\delta}_n] \\
\delta x_N &= \sum_{n=1}^{N} (\mathbf{n}_{n+1}^u \cdot \boldsymbol{\delta}_n) \\
\delta y_N &= \sum_{n=1}^{N} (\mathbf{o}_{n+1}^u \cdot \boldsymbol{\delta}_n) \\
\delta z_N &= \sum_{n=1}^{N} (\mathbf{a}_{n+1}^u \cdot \boldsymbol{\delta}_n)
\end{aligned}
\tag{2.50}
$$

where $\mathbf{n}_{n+1}^u$, $\mathbf{o}_{n+1}^u$, $\mathbf{a}_{n+1}^u$, and $\mathbf{p}_{n+1}^u$ are the unit vectors comprising the $\mathbf{U}_{n+1}$ matrix as given by

$$
\mathbf{U}_{n+1} = \begin{bmatrix} \mathbf{n}_{n+1}^u & \mathbf{o}_{n+1}^u & \mathbf{a}_{n+1}^u & \mathbf{p}_{n+1}^u \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.51}
$$

$$
\delta \mathbf{A}_n = \begin{bmatrix} \mathbf{d}_n \\ \boldsymbol{\delta}_n \end{bmatrix} \tag{2.52}
$$

It may be seen from 2.51 that $\mathbf{U}_{n+1}$ is a function of the nominal kinematics and joint angles since it is the product of $\mathbf{A}$ matrices. However, $\mathbf{d}_n$, $\boldsymbol{\delta}_n$ through Equations 2.31 and 2.32 are functions of the kinematic error parameters $\Delta\theta_n$, $\Delta\alpha_n$, Δl_n, Δr_n. Expanding the elements of the tool error terms in Equation 2.49 we see that

$$
\begin{aligned}
dx_N = \sum_{n=1}^{N} & [(\mathbf{n}_{n+1}^u \cdot \mathbf{k}_n^1) + (\mathbf{p}_{n+1}^u \times \mathbf{n}_{n+1}^u) \cdot \mathbf{k}_n^2] \Delta\theta_n + (\mathbf{n}_{n+1}^u \cdot \mathbf{k}_n^2) \Delta r_n \\
& + (\mathbf{n}_{n+1}^u \cdot \mathbf{k}_n^3) \Delta l_n + [(\mathbf{p}_{n+1}^u \times \mathbf{n}_{n+1}^u) \cdot \mathbf{k}_n^3] \Delta\alpha_n \\
dy_N = \sum_{n=1}^{N} & [(\mathbf{o}_{n+1}^u \cdot \mathbf{k}_n^1) + (\mathbf{p}_{n+1}^u \times \mathbf{o}_{n+1}^u) \cdot \mathbf{k}_n^2] \Delta\theta_n + (\mathbf{o}_{n+1}^u \cdot \mathbf{k}_n^2) \Delta r_n \\
& + (\mathbf{o}_{n+1}^u \cdot \mathbf{k}_n^3) \Delta l_n + [(\mathbf{p}_{n+1}^u \times \mathbf{o}_{n+1}^u) \cdot \mathbf{k}_n^3] \Delta\alpha_n \\
dz_N = \sum_{n=1}^{N} & [(\mathbf{a}_{n+1}^u \cdot \mathbf{k}_n^1) + (\mathbf{p}_{n+1}^u \times \mathbf{a}_{n+1}^u) \cdot \mathbf{k}_n^2] \Delta\theta_n + (\mathbf{a}_{n+1}^u \cdot \mathbf{k}_n^2) \Delta r_n \\
& + (\mathbf{a}_{n+1}^u \cdot \mathbf{k}_n^3) \Delta l_n + [(\mathbf{p}_{n+1}^u \times \mathbf{a}_{n+1}^u) \cdot \mathbf{k}_n^3] \Delta\alpha_n \\
\delta x_N = \sum_{n=1}^{N} & [(\mathbf{n}_{n+1}^u \cdot \mathbf{k}_n^2) \Delta\theta_n + (\mathbf{n}_{n+1}^u \cdot \mathbf{k}_n^3) \Delta\alpha_n] \\
\delta y_N = \sum_{n=1}^{N} & [(\mathbf{o}_{n+1}^u \cdot \mathbf{k}_n^2) \Delta\theta_n + (\mathbf{o}_{n+1}^u \cdot \mathbf{k}_n^3) \Delta\alpha_n] \\
\delta z_N = \sum_{n=1}^{N} & [(\mathbf{a}_{n+1}^u \cdot \mathbf{k}_n^2) \Delta\theta_n + (\mathbf{a}_{n+1}^u \cdot \mathbf{k}_n^3) \Delta\alpha_n]
\end{aligned}
\tag{2.53}
$$

Although computationally complex, the above result may be written as two linear equations

$$\mathbf{d}_N = \mathbf{m}_1\Delta\boldsymbol{\theta} + \mathbf{m}_2\Delta\mathbf{r} + \mathbf{m}_3\Delta\mathbf{l} + \mathbf{m}_4\Delta\boldsymbol{\alpha} \tag{2.54}$$

$$\boldsymbol{\delta}_N = \mathbf{m}_2\Delta\boldsymbol{\theta} + \mathbf{m}_3\Delta\boldsymbol{\alpha} \tag{2.55}$$

or by one equation

$$\begin{bmatrix}\mathbf{d}_N\\ \boldsymbol{\delta}_N\end{bmatrix} = \begin{bmatrix}\mathbf{m}_1\\ \mathbf{m}_2\end{bmatrix}\Delta\boldsymbol{\theta} + \begin{bmatrix}\mathbf{m}_2\\ 0\end{bmatrix}\Delta\mathbf{r} + \begin{bmatrix}\mathbf{m}_3\\ 0\end{bmatrix}\Delta\mathbf{l} + \begin{bmatrix}\mathbf{m}_4\\ \mathbf{m}_3\end{bmatrix}\Delta\boldsymbol{\alpha} \tag{2.56}$$

where $\mathbf{d}_N = [dx_N, dy_N, dz_N]^T$ are the three translational errors at the end of manipulator, $\boldsymbol{\delta}_N = [\delta x_N, \delta y_N, \delta z_N]^T$ are the three rotational errors at the end of manipulator, and $\Delta\boldsymbol{\theta}$, $\Delta\mathbf{r}$, $\Delta\mathbf{l}$, $\Delta\boldsymbol{\alpha}$ are $N \times 1$ column vectors of the kinematic error parameters.

Equation 2.56 may be formulated as a conventional Jacobian representation by

$$\delta\mathbf{T}_N = \mathbf{J}_K\delta\mathbf{k} \tag{2.57}$$

where $\delta\mathbf{k}$ is a $4N \times 1$ column vector of kinematic error parameters. Hence

$$\begin{bmatrix}\mathbf{d}_N\\ \boldsymbol{\delta}_N\end{bmatrix} = \begin{bmatrix}\mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{m}_4\\ \mathbf{m}_2 & 0 & 0 & \mathbf{m}_3\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{\theta}\\ \Delta\mathbf{r}\\ \Delta\mathbf{l}\\ \Delta\boldsymbol{\alpha}\end{bmatrix} \tag{2.58}$$

The Jacobian is a six row by $4N$ column matrix. Note that the column of $\mathbf{J}_K$ associated with $\Delta\boldsymbol{\theta}$, that is $(\mathbf{m}_1\,\mathbf{m}_2)^T$ may be identified as the joint angle Jacobian defined by Paul [22].

Equation 2.57 is the equation that relates small variations in the kinematic parameters, $\delta\mathbf{k}$, to variations in the end effector pose as given by $\delta\mathbf{T}_n$. The Jacobian, $\mathbf{J}_K$, is a function of the nominal kinematic parameters as well as the joint angles so the relationship may vary significantly over the workspace.

2.6 LIMITATIONS OF THE DENAVIT–HARTENBERG METHOD

Although the Denavit–Hartenberg model has been popular for modeling manipulator kinematics, several problems arise when using this model in a calibration procedure. As described in Section 2.3, the link coordinate frames are located at the intersection of the joint axis and the common normal. This implies that the location of these coordinate frames is a function of the manipulator geometry

and that small variations in this geometry will cause these frames to shift. This requirement leads to three effects that are undesirable for robot calibration:

- Selection of the reference or base frame is not arbitrary.
- The "zero position" of the manipulator is not arbitrary.
- Constants in the transformations vary by large amounts for revolute joints with nearly parallel axes.

The purpose of a robot calibration is to improve the mapping between joint space and task space so that objects in task space may be grasped or manipulated. This implies that the task space is well defined and that the task space location of both objects and the manipulator is precisely known. To locate the manipulator, the transformation between the robot base frame and the task space coordinate system must be accurate. To make this possible, some set of reference markers must be available so that the robot base frame may be easily related to the task space coordinate system. This may be accomplished by placing physical references such as tooling balls, reference planes, or locating pins on the robot base. To be effective, these references must either define the robot base frame or their location in the base frame must be known. Since the DH formalism restricts the location of the base frame to a joint axis, additional transformation parameters must be included to express the relationship between the joint 1 axis and the base coordinate system as defined by the reference marks. This seemingly obvious point is often overlooked when developing a calibration procedure. Precise knowledge of the kinematic parameters will do little good if the robot base frame cannot be located accurately in the task space coordinate system. A mechanism to accomplish this should be included in the calibration model.

As described in Section 2.3, the displacement of a revolute joint is given by the angle between the common normal and the x axis of the previous link coordinate system. The position of these joints when at zero displacement, therefore, is with the x axes of consecutive revolute joints aligned. This implies that the configuration of the manipulator in the zero position is dependent on the robot geometry. Small changes in the axis alignment will cause variations in the link coordinate systems and, hence, the zero position. One of the motivations behind manipulator calibration is the desire to have several, slightly different manipulators perform as though they were identical. If each manipulator has a different reference position, this goal is not met. Again, an extension to the DH formalism must be made so that an arbitrarily defined zero position is possible.

The final and most important limitation of the DH formalism is the treatment of consecutive revolute joints with nearly parallel axes. Many manipulators are designed to have revolute joints with parallel axes. In this case, there is no unique common normal and it is suggested that a common normal be chosen so that the coordinate frame is located in a convenient place. If, however, as part of the calibration procedure it is determined that the axes are not parallel but inclined to each other by a small amount, the common normal becomes unique and the location of the coordinate frame may change significantly. This discontinuous

and widely varying location of the coordinate frame causes the kinematic parameters in the DH model to change rapidly and over a large range. The problem may be analyzed by considering two consecutive frames F_1 and F_2 as shown in Figure 2.5. Here, frame 2 is slightly misaligned with frame 1 so that the unit vector $\mathbf{u}$ representing the direction of the joint axis is not aligned with z_1. Using F_1 as a reference frame we may write

$$\delta_x \mathbf{i} + \delta_y \mathbf{j} + c\mathbf{u} = r_2 \mathbf{k}_1 + \mathbf{a}_2 \tag{2.59}$$

Since the frames are misaligned, $\mathbf{a}_2$ denotes the unique common normal hence

$$\mathbf{a}_2 \cdot \mathbf{k}_1 = 0 \tag{2.60}$$

$$\mathbf{a}_2 \cdot \mathbf{u} = 0 \tag{2.61}$$

solving for the offset distance r_2 we get from Equations 2.59, 2.60, and 2.61

$$r_2 = \frac{\delta_x u_x u_z + \delta_z u_y u_z}{u_x^2 + u_y^2} \tag{2.62}$$

The above equation shows that when $\mathbf{u}$ is aligned with $z_1 (u_x = u_y = 0)$ the variation in the offset is discontinuous. Further, it may be shown that as the axes move from the nonparallel to the parallel situation the location of the common normal is discontinuous.

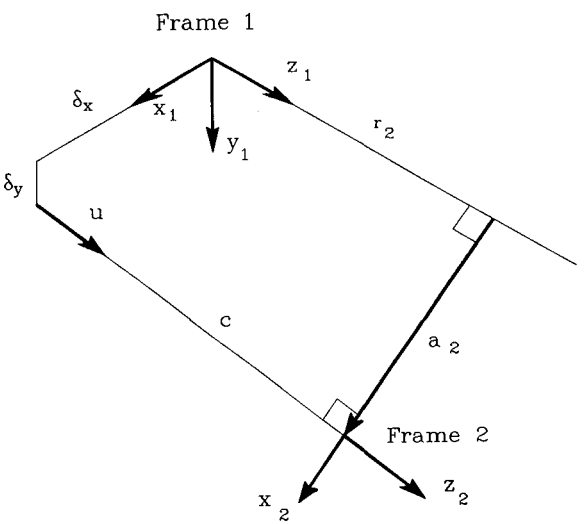


Figure 2.5. Joint axis misalignment.

It should be pointed out that this becomes a problem only in the selection of a model for parameter identification prior to robot calibration. If the nominal robot geometry indicates parallel axes we might be tempted to use a conventional Denavit–Hartenberg model and set the offset to zero. If in the process of calibration it is determined that the joints are misaligned, then r_2 will be nonzero and by a large amount. This discontinuous behavior may lead to convergence problems in the parameter identification routines (see Sections 4.3 and 4.4). If we assume a value for the offset and the joints turn out to be exactly parallel, then the offset becomes nonunique and again convergence may be compromised.

2.7 PROPERTIES OF A GOOD MODEL

Before proposing changes to the Denavit–Hartenberg formalism, the properties that a kinematic model should possess to make it suitable for calibration should be considered. Everett et al. [8] proposed that kinematic models for calibration should meet three criteria: completeness, proportionality, and equivalence. These concepts are described in the following paragraphs.

In classifying models, we will define a *complete* kinematic model as one that has the capability of relating the joint displacements to the tool pose for any manipulator while allowing for the arbitrary placement of the reference frame and arbitrary assignment of the zero position. Another way of defining completeness is to say that a complete model has enough coefficients to express any variation of the actual robot structure away from the nominal design. To be complete, the model must contain the required number of independent kinematic parameters. This section suggests a formula for computing the required number of independent parameters for a general manipulator.

Since kinematic identification is the process of finding a kinematic relationship between joint displacements and tool pose, it is necessary to establish a reference coordinate system (world frame) and a tool coordinate system (tool frame). The world frame should be fixed in a position so that measurements can be conveniently referenced to it. Typically, researchers locate the world frame on the fixed link of the manipulator, although it can be located anywhere. The tool frame should be conveniently fixed relative to the tool (last body in the chain). For both of these frames, convenience must be defined relative to the user of the manipulator. Requiring the tool frame to be located on a rotation axis should be avoided because although the axis may be well defined mathematically, it is difficult to specify or measure a pose relative to such an abstract and illusive feature.

After selecting the world and tool frames, joint coordinate frames are identified. Although not strictly necessary, it is convenient to use at least one frame for each joint. By orienting one axis of each joint frame in the direction of the motion, it is possible to express the joint's motion with a single unconstrained variable. This is due to the common assumption that the joints can be represented as lower pair mechanisms with a single and unique axis of motion. Furthermore this joint

variable is considered to be known or measurable through a feedback device. As a matter of convenience, compatibility with the literature, and without loss of generality, the z axis of each joint frame is assumed to be parallel with the joint axis of motion. Moreover, the origin of a revolute joint frame lies on the joint axis of motion.

The number of independent kinematic parameters is equal to the number of constraint equations required to completely specify the pose of the tool and joint frames. It is helpful to use simple examples when discussing constraint equations. Consider Figures 2.6 and 2.7, which depict single joint manipulators. Figure 2.6 shows a prismatic manipulator and Figure 2.7 shows a revolute. Both figures show arbitrarily located world and tool frames.

For both manipulator types, the orientation of the joint frame's z axis is assumed to be constant relative to the world frame. To ensure constant axis orientation requires two orientation constraint equations. Two additional constraints are required for the revolute joint manipulator to ensure the joint frame's origin lies on the rotation axis. This is not required for the prismatic joint.

Note that the origins of the joint frames are not completely constrained. Also note that the x axis of the joint frames need not be explicitly constrained. Because these constraints are missing one might expect that position feedback devices must somehow be adjusted to reflect the orientation and origin of the joint axis, but this is not always the case [17].

To proceed counting constraint equations, consider the tool frame. Recall the tool and world frames were arbitarily chosen to enable convenient measurement, hence the model must honor the chosen tool frame orientation. Because

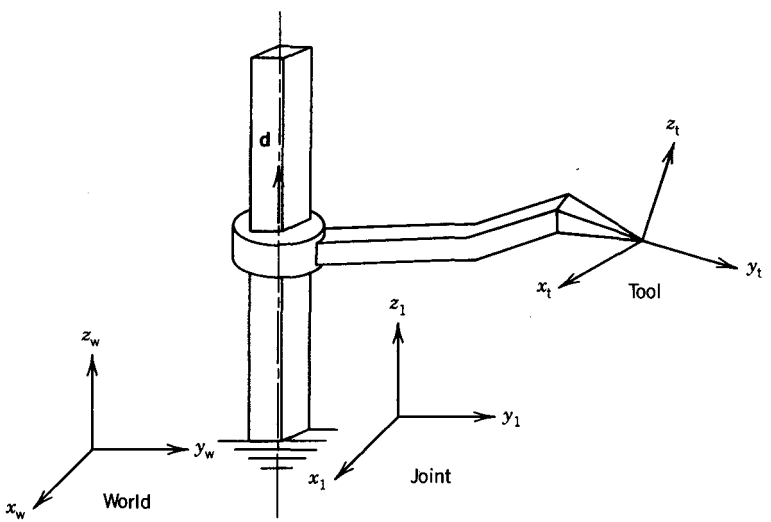


Figure 2.6. Prismatic manipulator.

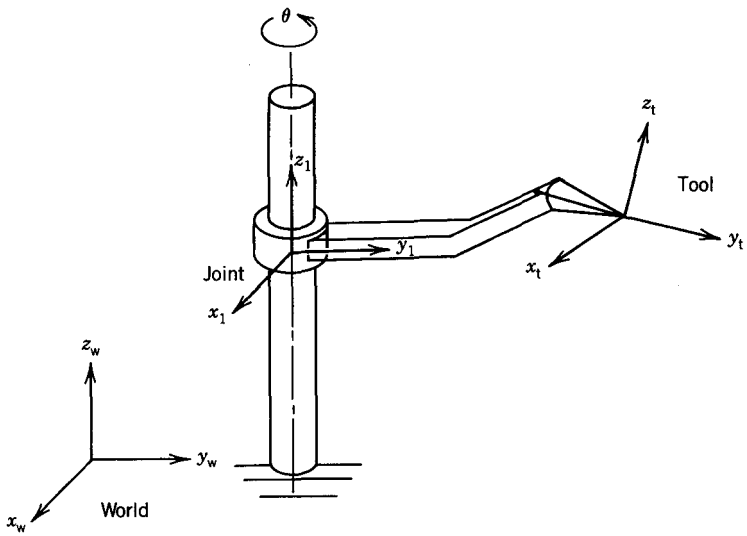


Figure 2.7. Revolute manipulator.

the orientation of the tool frame relative to the joint frame is a constant and arbitrary, three orientation constraints must be specified. In addition to the tool's orientation, its origin is completely constrained relative to the joint axis. Hence a total of six constraints are required to specify the tool frame relative to the joint frame.

In summary, the prismatic manipulator requires 8 constraint equations and the revolute requires 10 equations. Because the constraint equations can be represented with a single constant, there are respectively 8 and 10 independent kinematic parameters specifying the prismatic and revolute manipulators shown in Figures 2.6 and 2.7.

This analysis of constraints may be extended to serial link manipulators with multiple joints. The above argument leads to the conclusion that the number of constraints on a revolute joint indicates that four kinematic parameters are necessary and the two parameters are necessary for a prismatic joint. Also, six additional parameters are necessary to ensure independent location of the tool frame. This leads to the following equation for determining the number of parameters necessary for completeness.

$$N = 4R + 2P + 6 \tag{2.63}$$

where N is the required number of independent parameters, R is the number of revolute joints, and P is the number of prismatic joints. For a PUMA 560 manipulator, this equation indicates that a complete kinematic model should

contain 30 independent parameters. A standard DH model of the PUMA 560 contains 18 parameters. If we add six joint offsets so that an arbitrary zero position is possible, the total comes to 24. The final six parameters are added to allow arbitrary location of the tool frame. Equation 2.63 indicates that a complete model for a spherical manipulator such as the Stanford arm should have 28 parameters, which is verified by further analysis. It is important to note that Equation 2.63 refers to independent parameters. Additional parameters in a model will not gain completeness if any of the parameters are dependent. This may be determined by ensuring the the rank of the Jacobian as developed in Section 2.5 is equal to the number of model parameters.

A second property that a model should possess is *proportionality*. Proportionality implies that small changes in the robot structure should be reflected by small changes in the parameters in the kinematic model. As shown in Section 2.6, the DH model can yield widely varying model parameters for very small deviations in axis alignment for revolute joints with nearly parallel axes. Models that do not exhibit proportionality tend to produce numerical difficulties during the identification step. Many investigators have found that it is virtually impossible to determine the DH parameters for nearly parallel axes. This has led to a number of suggested modifications to the DH procedure which will be described in the following section.

Model *equivalence* refers to the ability to transform parameters of one model into parameters of another model. Any two complete models are necessarily equivalent. It follows that equivalence prevents one complete model from producing greater accuracy than another.

2.8 MODEL REVIEW

There are a number of ways to develop models that exhibit completeness, proportionality, and equivalence. In this section, we will review models proposed by a number of researchers. For convenience, we will collect these approaches into several categories: modifications of the DH method, the zero reference method, the single joint method, models for closed loop manipulators, and models for manipulators having joints with higher pairs.

2.8.1 Modifications of the Denavit–Hartenberg Method

A number of investigators have determined the limitations of the DH model and have taken steps to modify the modeling procedure. Most of the work has centered around techniques of modifying the model for consecutive revolute joints with parallel axes. In this section, we will present a modification to the DH model that has the properties of completeness, proportionality, and equivalence. This will be followed by a review of the current literature and a brief description of the various models that have been proposed.

2.8.1.1 A Modified Denavit–Hartenberg Model The first modification to the DH approach will be to gain proportionality for revolute joints. This may be accomplished by modifying the standard DH transformation. The following development follows the one proposed by Hayati and Mirmirani [12]. Assume that we have two consecutive revolute joints with axes n and $n+1$ as shown in Figure 2.8. Rather than using the common normal, we will define a plane that is perpendicular to the joint n axis and that passes through the origin of the $n-1$ coordinate system, O_{n-1}. The intersection of this plane with the joint $n+1$ axis defines the origin of the n coordinate system, point O_n. The line drawn between point O_{n-1} and point O_n defines the direction of the x axis for the n coordinate system. The z axis lies along the joint $n+1$ axis. Given these definitions, the transformation between the n and $n-1$ coordinate axes is written as

$$\mathbf{T}_n = \mathbf{R}(z, \theta_n)\mathbf{T}(r_n, 0, 0)\mathbf{R}(x, \alpha_n)\mathbf{R}(y, \beta_n) \tag{2.64}$$

where r_n is the length of the line between O_{n-1} and O_n and α_n and β_n are rotations about the indicated axes to align the z axis with the joint axis. The variable θ_n is the joint variable. When the series of transformations in Equation 2.64 is expanded, the following transformation results:

$$\mathbf{T}_n = \begin{bmatrix} -s\alpha_n s\beta_n s\theta_n + c\beta_n c\theta_n & -c\alpha_n s\theta_n & s\alpha_n c\beta_n s\theta_n + s\beta_n c\theta_n & r_n c\theta_n \\ s\alpha_n s\beta_n c\theta_n + c\beta_n s\theta_n & c\alpha_n c\theta_n & -s\alpha_n c\beta_n c\theta_n + s\beta_n s\theta_n & r_n s\theta_n \\ -c\alpha_n s\beta_n & s\alpha_n & c\alpha_n c\beta_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.65}$$

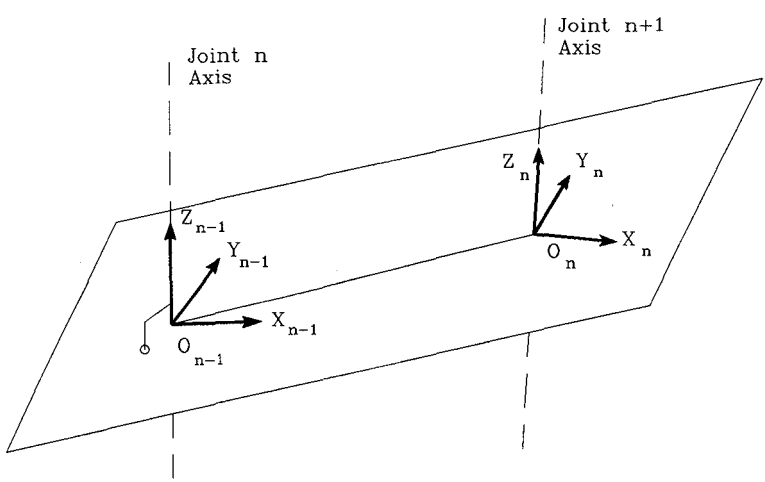


Figure 2.8. Nearly parallel revolute axes.

where $s\alpha$ represents $\sin\alpha$, $c\alpha$ represents $\cos\alpha$, and so on. This transformation may be used for consecutive revolute joints with nearly parallel axes.

The elimination of reliance on the common normal means that small deviations in axis orientation will produce proportional changes in the parameters for parallel axes. To continue the discussion of modifications to the DH approach, it is useful to consider an example case. We will develop a model for the 3 DOF manipulator shown in Figure 2.9. As discussed earlier, it is important that the base frame be located in an arbitrary position. To accomplish this, we will define frame x_B, y_B, z_B to be the base frame located in the position shown in Figure 2.9. Note that this frame is not required in the standard DH procedure and has an arbitrary location. Frame 0 is located on joint axis 1 as defined by the DH rules. Since the z axes of the base frame and frame 0 are nearly aligned, we will use the transformation for parallel axes given in Equation 2.65 to relate the two frames. Again, it is not necessary to align the z axes of the B and 0 coordinate systems.

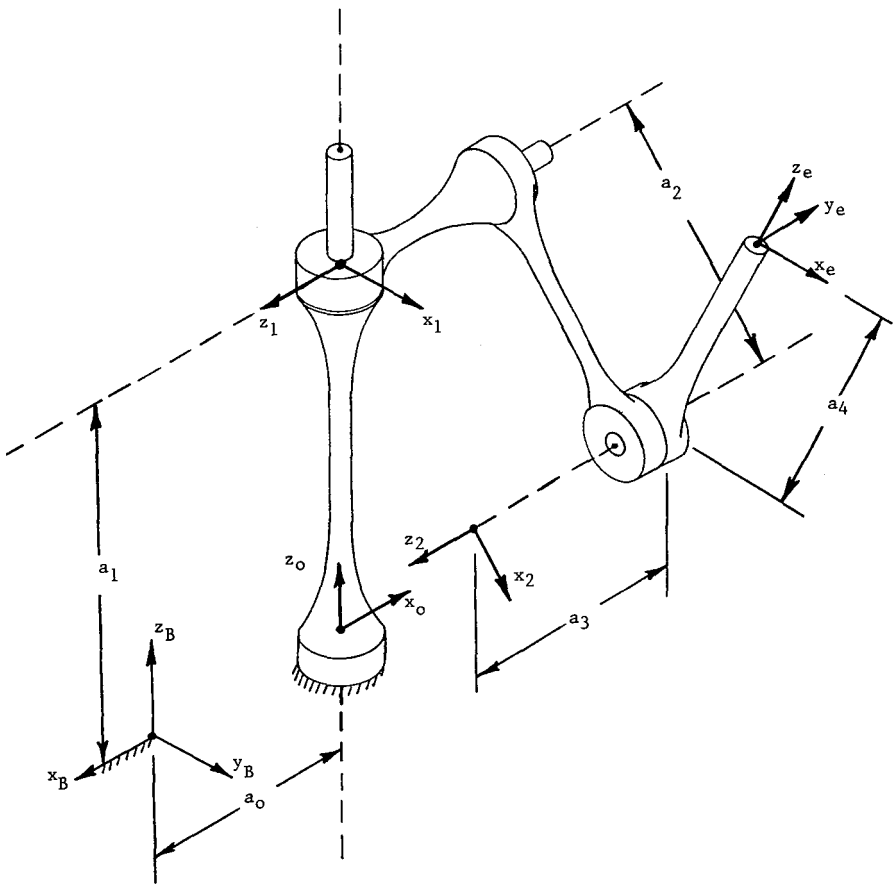


Figure 2.9. Modified DH model for 3 DOF manipulator.

If these axes are not aligned, the standard DH transformation may be used. Frames 1 and 2 are located on the appropriate joint axes. The transformations between frames are standard DH except for the transformation between 2 and 1, which are nearly parallel axes. The final frame to be located is the tool frame, *e*. As described in Section 2.7, the position and orientation of this frame must also be arbitrary. To accomplish this, the location of frame *e* with respect to frame 2 will be given by the following series of transformations:

$$\mathbf{T}_4 = \mathbf{R}(z, \theta_3)\mathbf{R}(y, \beta)\mathbf{R}(x, \alpha)\mathbf{T}(x, dx)\mathbf{T}(y, dy)\mathbf{T}(z, dz) \tag{2.66}$$

where θ_3 is the joint displacement for axis 3, α and β are rotations about the x and y axes, and dx, dy, and dz are displacements along the indicated coordinate axes. Multiplication of the transformations indicated in Equation 2.66 results in the following transformation between frame *e* and frame 2.

$$\mathbf{T}_4 = \begin{bmatrix} c\beta c\theta_3 & -c\alpha s\theta_3 + s\alpha s\beta c\theta_3 & s\alpha s\theta_3 + c\alpha s\beta c\theta_3 & c_{14} \\ c\beta s\theta_3 & c\alpha c\theta_3 + s\alpha s\beta s\theta_3 & -s\alpha c\theta_3 + c\alpha s\beta s\theta_3 & c_{24} \\ -s\beta & s\alpha c\beta & c\alpha c\beta & c_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.67}$$

where

$$c_{14} = dx(c\beta c\theta_3) + dy(-c\alpha s\theta_3 + s\alpha s\beta c\theta_3) + dz(s\alpha s\theta_3 + c\alpha s\beta c\theta_3) \tag{2.68}$$

$$c_{24} = dx(c\beta s\theta_3) + dy(c\alpha c\theta_3 + s\alpha s\beta s\theta_3) + dz(-s\alpha c\theta_3 + c\alpha s\beta s\theta_3) \tag{2.69}$$

$$c_{34} = -dxs\beta + dys\alpha c\beta + dzc\alpha c\beta \tag{2.70}$$

As shown in the equations, this transformation has five constants and one joint variable. To summarize, there are five coordinate systems used in the model. These are the base frame (*B*), frames attached to each link (0 through 2), and the end effector frame *e*. Transformations between each link have been chosen to ensure that proportionality is maintained and the end effector transformation has been specified so that there is no reliance on robot geometry for the location of the end effector system. The transformation between the *B* and the 0 frames contains four constant parameters. Each of the next two transformations (0 through 2) contains three constant parameters and one joint variable. The final transformation (2 to *e*) consists of five parameters and one joint variable. This leaves us with a total of 15 constant parameters and 3 joint variables. If we apply Equation 2.63, we find that a total of 18 parameters is necessary to ensure completeness. A look back at the model will indicate that, as currently defined, the zero position of the manipulator is dependent on the robot geometry. If the orientation of the axes changes, the alignment of the common normals and, hence, the zero position will shift. This may be addressed by adding a constant

joint offset to each joint variable. Each joint variable in the various transformations would be replaced with the following expression

$$\theta_i = \theta_i' + \Delta\theta_i \tag{2.71}$$

where θ_i' is the joint variable reported by the robot controller and $\Delta\theta_i$ is the constant offset. Note that this has the effect of including a level 1 calibration in the kinematic model and brings the number of constant parameters in the model up 18 as indicated by Equation 2.63. The model is now complete and the remaining steps of the calibration process (measurement, identification, and correction) may be addressed.

2.8.1.2 Literature Review A number of investigators have proposed modifications to the DH model that address proportionality or completeness. Although these various approaches differ in detail, the effects are similar to the procedure described above. The definition of the "most desirable" approach is usually a strong function of an individual's background or personal preference. The following review of the literature is included for those who may be interested in alternate model formulations.

One of the first authors to address the robot calibration problem was Wu [34, 35]. In these papers, a calibration model was developed but no effort was made to use the model in an identification procedure and the limitations of the model did not become apparent. In later years, other authors such as Ibarra and Perriera [18], Zhen [36], and Payannet, Aldon, and Liegeois [23] published calibration models that followed the standard DH formalism. In 1983, works by Mooring [20] and Hayati [13] pointed out the proportionality problems inherent in the standard DH approach. At that time, Hayati [13] proposed a modification to the DH formalism similar to the one in the example given above for parallel revolute axes. A number of other authors such as Judd and Knasinski [19], Puskorius and Feldkamp [24], Sugimoto and Okada [28], and Hollerbach and Bennett [15, 16] subsequently reported the use of similar modified DH models. While the modified DH model proposed by Hayati for revolute joints contained the proper number of parameters for completeness (4), other authors proposed joint models that contained five or even six parameters to describe a revolute axis. Since the inclusion of additional parameters leads to singularities in the Jacobian, some of these additional parameters must be specified or eliminated through some numerical procedure before the identification process can be completed. Examples of models containing such additional parameters are Hsu and Everett [17], Veitschegger and Wu [31, 32], Chen and Chao [2], Stone, Sanderson, and Neuman [25–27], Driels and Pathre [6, 7], and Whitney, Lozinski, and Rourke [33]. Vaishnav and Magrab [30] in 1987 proposed a nine parameter model that allowed for nonorthogonal coordinate systems to be represented. An excellent and more in-depth survey of kinematic models for robot calibration has been published by Hollerbach [14].

When considering the various kinematic models that have been proposed for calibration, one should carefully consider the number of parameters available in each model. If identification of robot kinematics is the only goal of the calibration procedure, the formula for completeness, Equation 2.63, will indicate the proper number of parameters for the model. Inclusion of fewer parameters may lead to an incomplete identification and the use of more parameters can lead to numerical difficulties in the identification phase. Of course, the inclusion of nonkinematic or nongeometric parameters in the model can lead to more parameters than indicated by Equation 2.63. One should be careful, however, to be sure that additional parameters deal solely with the nongeometric properties and are independent of the kinematic parameters.

2.8.2 Zero-Reference Model

An approach to kinematic modeling that does not rely on the Hartenberg–Denavit formalism has been proposed for use in manipulator calibration by Mooring [20]. This procedure is based on Rodrigues equation and consists of establishing a reference coordinate system that is fixed in the work space and an end effector coordinate system that is attached to the end effector of the robot. The orientation of the individual joint axes is determined by locating a unit vector on each axis that defines the direction of each axis. The location of each of the axes is determined by defining a point through which the axis passes. The unit vectors and points are specified in the reference coordinate system.

To further illustrate this approach, we will begin by considering a single revolute joint. Figure 2.10 shows a link that is constrained to rotate about the revolute joint. The joint axis is indicated with a dashed line and the reference coordinate system is also shown in the figure. The unit vector **u** is defined to lie along the joint axis so that positive rotation is defined by the direction of **u** and the right hand rule. The point **p** is defined to be any point that lies along the joint axis. It may be shown that the location of any point **r** on the link after a rotation will be given by

$$\mathbf{r} = \mathbf{D}\mathbf{r}' = \begin{bmatrix} & \mathbf{R} & & (\mathbf{I}-\mathbf{R})\mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{r}' \tag{2.72}$$

where **I** is the identity matrix, $\mathbf{r}'$ is the location of **r** before the rotation, and **R** is given by

$$\mathbf{R} = \begin{bmatrix} u_x^2 v\phi + c\phi & u_x u_y v\phi - u_z s\phi & u_x u_z v\phi + u_y s\phi \\ u_x u_y v\phi + u_z s\phi & u_y^2 v\phi + c\phi & u_y u_z v\phi - u_x s\phi \\ u_x u_z v\phi - u_y s\phi & u_y u_z v\phi + u_x s\phi & u_z^2 v\phi + c\phi \end{bmatrix} \tag{2.73}$$

where u_x, u_y, and u_z are the components of **u** and $s\phi$ implies $\sin\phi$, $c\phi$ implies $\cos\phi$,

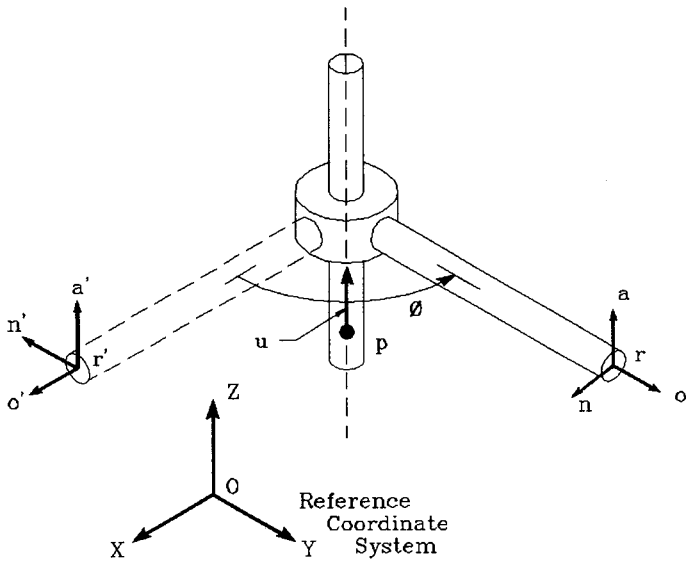


Figure 2.10. Zero reference position method.

and $v\phi$ implies vers ϕ or $1 - \cos\phi$. In other words, to specify a rotation, one must specify the unit vector **u** that defines the axis orientation, the point **p** that locates the axis in space, and the angle of rotation, ϕ. A more detailed description and derivation of Equations 2.72 and 2.73 are given by Suh and Radcliffe [29]. It is important to note that of the three components of **u**, only two are independent since the vector is of unit length. Also, since the point **p** may be any point on the axis of rotation, it also has only two independent components. In other words, if two components of **p** are specified, the third may be determined since it must satisfy the equation of the line that defines the axis of motion. There are, therefore, five independent quantities necessary to define the displacement matrix **D** for a revolute joint.

To model a prismatic joint, the unit vector **u** is still used to define the axis orientation but the location of the axis is not required. Since a prismatic joint generates pure translation, only the direction of motion is significant. The displacement would, therefore, be given by

$$\mathbf{r} = \mathbf{D}\mathbf{r}' = \begin{bmatrix} 1 & 0 & 0 & su_x \\ 0 & 1 & 0 & su_y \\ 0 & 0 & 1 & su_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{r}' \tag{2.74}$$

where s is the joint displacement. Since **u** is still a unit vector, it has only two

independent values. A total of three independent values, therefore, are required to define the displacement matrix **D** for a prismatic axis.

The displacement matrices described above may be used to model a manipulator in the following manner. First, a reference coordinate frame is established. The location and orientation of this frame is arbitrary. Next, a zero position for the manipulator is defined. This is simply the position that we wish the robot to be in when all the joint displacements are zero. It is interesting to note that the Hartenberg–Denavit procedure does not allow freedom in the selection of the zero position. The values of **u** and **p** are then defined for every joint while the robot is in the zero position. Finally, a coordinate system is located at any convenient position on the end effector and the transformation between the end

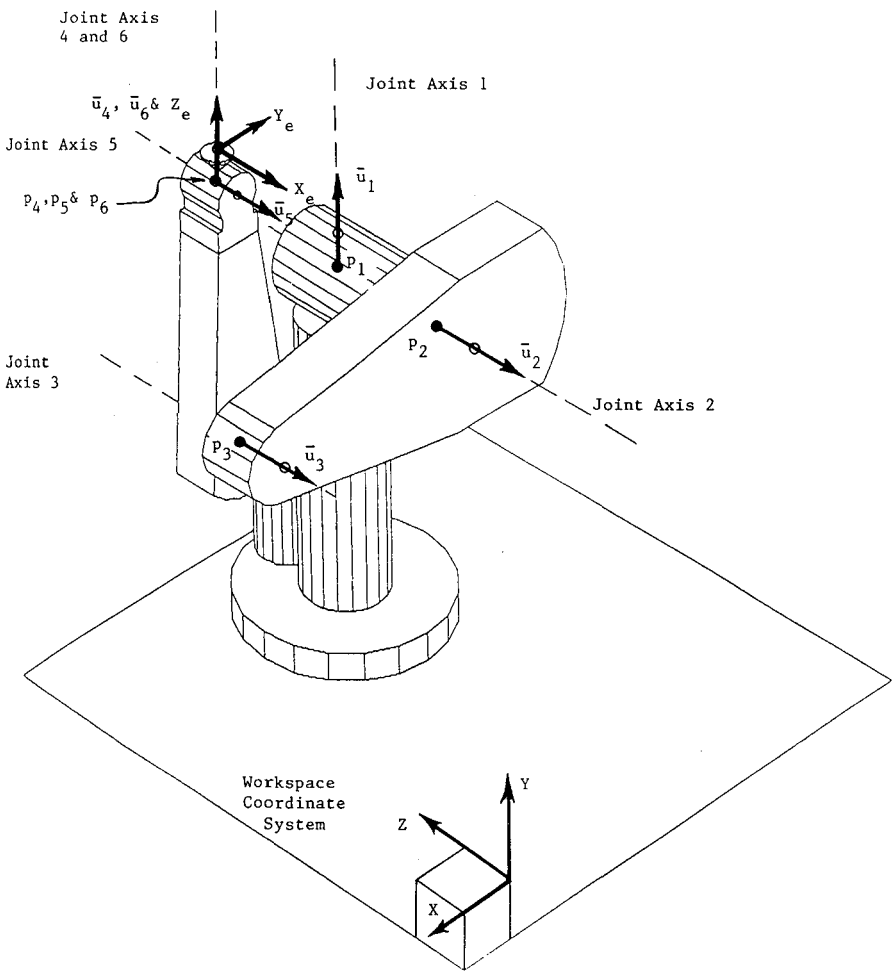


Figure 2.11. PUMA 560—zero reference position method.

TABLE 2.1. PUMA 560—Zero Reference Position Parameters

Joint	u_x	u_y	u_z	p_x	p_y	p_z
1	0	1	0	−381.00	373.38	853.44
2	0	0	−1	−381.00	373.38	704.35
3	0	0	−1	50.85	373.38	704.35
4	0	1	0	30.52	806.38	704.35
5	0	0	−1	30.52	806.38	704.35
6	0	1	0	30.52	806.38	704.35

effector system is defined. This is also done while the robot is in the zero position. Since the location of the end effector coordinate system is arbitrary, there are six independent quantities in this transformation.

We have now defined all of the elements of each joint displacement matrix except the actual joint motions. For any combination of joint displacements, the transformation from the end effector system to the reference frame will be given by

$$^{r}\mathbf{T} = \mathbf{D}_1\mathbf{D}_2\mathbf{D}_3\mathbf{D}_4\mathbf{D}_5\mathbf{D}_6\mathbf{T}_0 \tag{2.75}$$

where $\mathbf{D}_i$ represents the displacement matrix for the ith joint and $\mathbf{T}_0$ represents the transformation between the end effector frame and the reference frame in the zero position.

As an example of this method, we will consider the PUMA 560 manipulator illustrated in Figure 2.11. Note that the reference coordinate system has been located away from the base of the manipulator in a position that is convenient for defining objects in the workspace. The choice of position and orientation for this frame is arbitrary. The robot is shown in the zero position and the values for the unit vectors $\mathbf{u}_i$ and points $\mathbf{p}_i$ are listed in Table 2.1. Note that both the unit vectors $\mathbf{u}_i$ and the points $\mathbf{p}_i$ are defined with respect to the reference coordinate frame. The location of $\mathbf{p}_1$ must be determined with some measuring device so that it is known precisely in the reference coordinate system. The location of the other points $\mathbf{p}_i$ may be determined from knowledge of the robot geometry. Given the information in Table 2.1 and a set of joint displacements, the displacement matrices $\mathbf{D}_1$ through $\mathbf{D}_6$ may be defined for the manipulator. To complete the pose description, the transformation from the end effector frame to the reference frame must be defined for the manipulator when it is in the zero position. As shown in Figure 2.11, this transformation may be written by inspection and is given by

$$\mathbf{T}_0 = \begin{bmatrix} 0 & -1 & 0 & 30.52 \\ 0 & 0 & 1 & 862.58 \\ -1 & 0 & 0 & 704.35 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.76}$$

This completes the description of the manipulator model if the manipulator has no structural error. If we wish to use this model for calibration, however, the appropriate model parameters must be identified for each joint axis. In the case of a revolute joint, the components of the unit vector **u** and the point **p** must be defined. As stated earlier, there are four independent values associated with these vectors. To ensure the property of proportionality, it is important to choose the proper components of the vectors to be the "unknown" coefficients in the model. This is easily done since most commercially available robot manipulators have consecutive joint axes that are parallel or perpendicular. When in the zero position, these axes nearly line up with coordinate axes. As shown in Figure 2.11, the axes of a perfect PUMA manipulator are parallel with the Y or Z axes of the reference frame in the zero position. This fact may be used to select the components of the unit vector **u** to be used in the calibration model. To illustrate this, we will consider a joint axis that is nearly parallel to the Y coordinate axis as shown in Figure 2.12. As the joint axis orientation is varied about the nominal position, the u_x and u_z components of **u** vary in proportion to the degree of misalignment. The u_y component, however, shows very little change until the misalignment is significant. If we choose u_x and u_z as two of our joint parameters, the component u_y will be given by

$$u_y = \sqrt{1 - u_x^2 - u_z^2} \tag{2.77}$$

In a similar manner, we may choose the components of the point **p** to use in the model. If the axis is nearly parallel to the Y axis as shown in Figure 2.12, variations in location of the intersection of the actual joint axis with a plane perpendicular to the Y axis will be proportional to the variations in axis location.

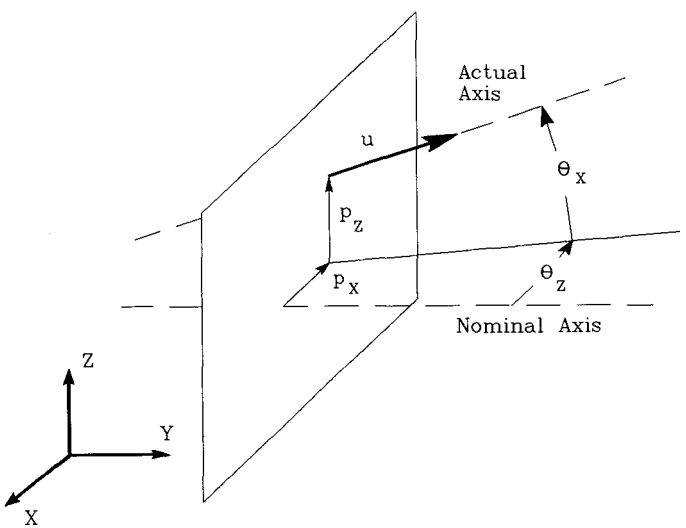


Figure 2.12. Joint axis nearly parallel to Y axis.

TABLE 2.2. PUMA 560—Zero Reference Position Parameters for Calibration

Joint	u_x	u_y	u_z	p_x	p_y	p_z
1	u_{1x}	$\sqrt{1-u_{1x}^2-u_{1z}^2}$	u_{1z}	$p_{1x}-381.00$	373.38	$p_{1z}+853.44$
2	u_{2x}	u_{2y}	$-\sqrt{1-u_{2x}^2-u_{2y}^2}$	$p_{2x}-381.00$	$p_{2y}+373.38$	704.35
3	u_{3x}	u_{3y}	$-\sqrt{1-u_{3x}^2-u_{3y}^2}$	$p_{3x}-50.85$	$p_{3y}+373.38$	704.35
4	u_{4x}	$\sqrt{1-u_{4x}^2-u_{4z}^2}$	u_{4z}	$p_{4x}-30.52$	806.38	$p_{4z}+704.35$
5	u_{5x}	u_{5y}	$-\sqrt{1-u_{5x}^2-u_{5y}^2}$	$p_{5x}-30.52$	$p_{5y}+806.38$	704.35
6	u_{6x}	$\sqrt{1-u_{6x}^2-u_{6z}^2}$	u_{6z}	$p_{6x}-30.52$	806.38	$p_{6z}+704.35$

We may choose to fix the p_y component to some convenient value and let p_x and p_z vary to describe the axis location. Note that setting p_y equal to a constant defines a plane perpendicular to the Y axis.

To illustrate the use of the zero reference approach for calibration, we will return to the example of the PUMA 560. Table 2.2 shows the definition of the **u** and **p** vectors for calibration. As shown in the table, there are 12 parameters associated with the **u** vectors and 12 parameters associated with the points **p**, which give a total of 24 parameters. As stated earlier, however, a complete PUMA model should have 30 parameters. The additional six parameters in this model are expressed in the joint offsets. There are two ways of incorporating these parameters. The first is to simply include an offset angle $\delta\phi_i$ to each joint displacement and then treat these six offsets as parameters in the model. In this case the rotation partition of the displacement matrix would be given by

$$\mathbf{R} = \begin{bmatrix} u_x^2 v\phi' + c\phi' & u_x u_y v\phi' - u_z s\phi' & u_x u_z v\phi' + u_y s\phi' \\ u_x u_y v\phi' + u_z s\phi' & u_y^2 v\phi' + c\phi' & u_y u_z v\phi' - u_x s\phi' \\ u_x u_z v\phi' - u_y s\phi' & u_y u_z v\phi' + u_x s\phi' & u_z^2 v\phi' + c\phi' \end{bmatrix} \tag{2.78}$$

where ϕ' is given by $\phi + \delta\phi$. With this approach, the relationship between the end effector and the reference coordinate frame when the robot is in the "zero position" will be very close to the same value for all of the calibrated robots. The joint offsets will allow for small rotations of the joints away from the zero position to compensate for the variations in the robot structure.

A second approach to making the model complete is to allow for variations in the $\mathbf{T}_0$ matrix. Recalling that $\mathbf{T}_0$ is the transformation between the end effector frame and the reference frame when the robot is in the zero position, there will be a total of six independent parameters (three rotation and three translation) necessary to define this matrix. If these parameters are treated as model parameters rather than constants, we will again have the required 30 parameters. This approach will always ensure that the joint displacements are zero in the "zero position" but the end effector may be in different positions for different manipulators after calibration. For most tasks, this approach is less desirable than the use of joint offsets.

In summary, the zero reference model is an alternative approach to modeling that does not rely on the Hartenberg–Denavit formalism. The approach yields a model that is complete, proportional, nonsingular for most robot configurations, and may be converted to an equivalent Hartenberg–Denavit model after calibration if necessary. For more information on this modeling approach, refer to the work by Mooring and Tang [20, 21].

2.8.3 Single Joint Method

Several investigators [20, 25] have proposed manipulator calibration procedures that approach the problem by considering the robot one joint at a time. Stone

[25–27] has used this approach and a unique data acquisition system to calibrate a series of PUMA 560 robots. With this method, joints 1 through 5 are fixed throughout a data collection session while joint 6 is studied. The parameters describing the motion for joint 6 are then determined and stored before moving to another joint. Since the mechanism is treated one joint at a time, the modeling process is quite different than the others described in this chapter. Because the model in this case is so closely related to the identification procedure, the details of the modeling procedure are given in Chapter 4 along with the description of the identification procedure.

2.8.4 Manipulators with Closed Loops

In the previous sections of this chapter, we considered only those manipulators that are made of open kinematic chains. In other words, one may progress from the base of the manipulator to the end effector by sequentially moving from one link to a joint and then to another link without retracing any path. In this section, we wish to consider the modeling of a manipulator with one closed kinematic loop. An example of such a robot is shown in Figure 2.13. The existence of a closed-loop kinematic chain in the mechanism adds several new aspects to the modeling process. The first is the existence of a number of dependent parameters in the model. The relationship between these parameters is determined by a loop

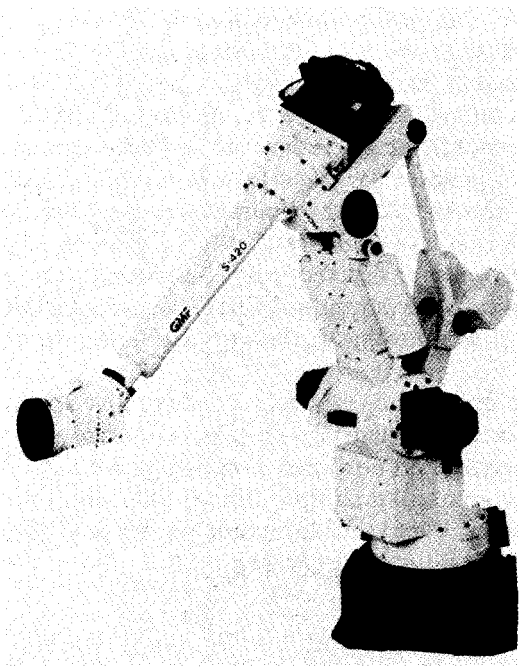


Figure 2.13. Manipulator with a closed kinematic chain. Photograph courtesy of GMFanuc Robotics, Auburn Hills, MI.

constraint equation, which exists in addition to the equation relating the end effector pose to the base frame. Another aspect of closed-loop robots is the existence of manipulator joints that do not have a prime mover or driving device. In a typical serial link manipulator, each joint has a motor or some actuator along with an associated feedback device. Closed loop robots are usually designed so that some of the joints have neither an actuator nor a transducer. Because these "passive" joints do not require a unique axis of motion, they can be comprised of higher kinematic pairs such as spherical joints. These aspects of closed loop robots combine to complicate the modeling process. The following discussion gives several examples of the generation of models for manipulators with closed loops.

According to Everett and Lin [9], two types of kinematic equations are required for a manipulator with a closed loop: (1) the open-loop transformation $\mathbf{T}_o$, which relates the end effector location relative to the world coordinate frame, and (2) the closed-loop transformation $\mathbf{T}_c$, which contains the closed-loop constraint equations.

A simple example is shown in Figure 2.14. The driving motors are located at joints 1 and 5, and the dependent variables are located at joints 2, 3, and 4. The $\mathbf{T}_o$ and $\mathbf{T}_c$ in this case may be expressed as

$$\mathbf{T}_o = \mathbf{A}_{w1}\mathbf{A}_{12}\mathbf{A}_{2t} \tag{2.79}$$

$$\mathbf{T}_c = \mathbf{A}_{12}\mathbf{A}_{23}\mathbf{A}_{34}\mathbf{A}_{45}\mathbf{A}_{51} = \mathbf{I} \tag{2.80}$$

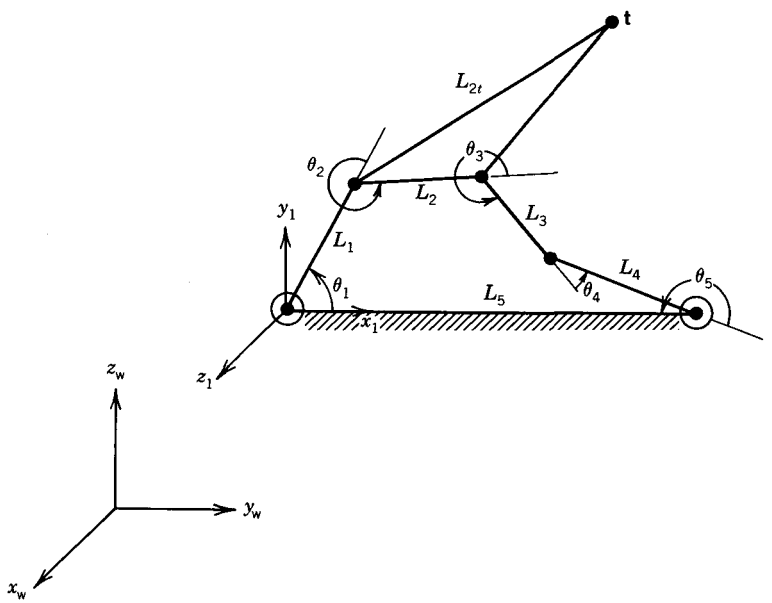


Figure 2.14. A simple closed-loop robot with a 2D constraint.

Here $\mathbf{A}_{w1}$ is the homogeneous transformation from the world frame to joint 1, and $\mathbf{A}_{12}$ is the homogeneous transformation from joint 1 to joint 2, etc. The matrix $\mathbf{I}$ is the identity matrix.

For each set of joint encoder readings, Equation 2.80 is used to calculate the dependent variables. These dependent variables are then used to calculate the $\mathbf{T}_o$ matrix. For example, to compute $\mathbf{T}_o$ in Equation 2.79, the value of θ_2 has to be calculated. Moreover, it is interesting to see that the transformation matrix $\mathbf{A}_{12}$ is used to calculate both $\mathbf{T}_o$ and $\mathbf{T}_c$.

From Equations 2.79 and 2.80, one can see the major difference between an open-loop robot and a closed-loop robot kinematic transformation. In a closed-loop transformation, some of the joint variables are dependent. As these dependent variables can be calculated by using the constraint equations, one can have fewer independent parameters in a closed-loop mechanism than in one with an open-loop geometry.

As some of the joints in a closed-loop robot do not have a driving device or a measured angle, a complete kinematic transformation for a closed-loop robot is more difficult to obtain than that of an open-loop robot. By using the rules stated in this section and the concept of closed-loop constraint equations, however, one may pick the independent parameters for a closed-loop robot. The following four examples are used to demonstrate model development.

2.8.4.1 A 4R Mechanism Figure 2.15 shows the number of parameters necessary in a 4R mechanism. The double arrowhead in the figure shows the direction of transformation from one joint to the next joint. The characters 2t and 2r represent two translational and two orientational parameters, respec-

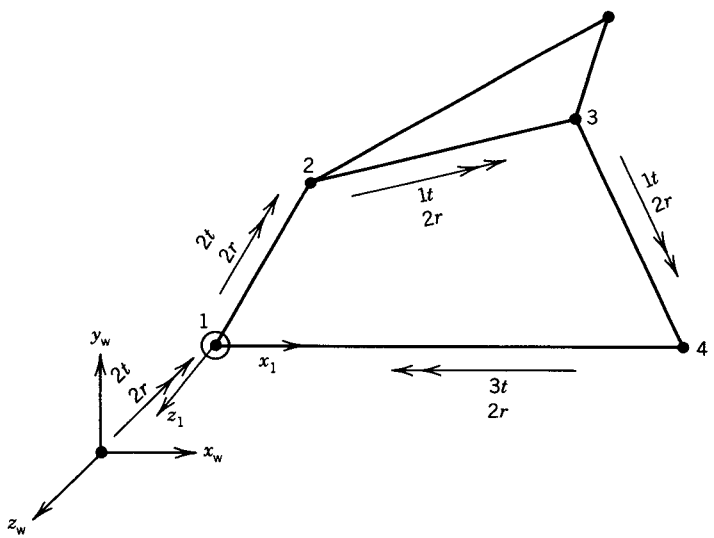


Figure 2.15. A 4R mechanism.

tively. Since this is a mechanism with one degree of freedom, joint 1 is assumed to have the prime mover and associated joint transducer. This is indicated in the figure by placing a circle around the joint. All joints are revolute joints whose axes are nominally in the Z_w direction.

To transform from the world coodinate to joint 1 and from joint 1 to joint 2 in Figure 2.15, four independent parameters for each transformation are needed. If the encoder rotates about the "Z_w" axis, the following parameters can be used to transform from joint 1 to joint 2:

$$\mathbf{R}(x, \theta_{1x})\mathbf{R}(y, \theta_{1y})\mathbf{R}(z, \theta_{1z})\mathbf{T}(x, l_1) \tag{2.81}$$

Similarly, to transform from joint 2 to joint 3, one may also use the following four parameters to complete the transformation:

$$\mathbf{R}(z, \theta_{2z})\mathbf{T}(x, l_2)\mathbf{R}(x, \theta_{2x})\mathbf{R}(y, \theta_{2y}) \tag{2.82}$$

Joint 2, however, does not have a prime mover and the value of θ_{2z} can be obtained from Equation 2.80. Only three parameters, therefore, are needed to define the position and orientation of the joint 3 axis with respect to joint 2.

Similary, to define the relationship between joint 4 and joint 3, only three parameters are required for the transformation

$$\mathbf{R}(z, \theta_{3z})\mathbf{T}(x, l_3)\mathbf{R}(x, \theta_{3x})\mathbf{R}(y, \theta_{3y}) \tag{2.83}$$

since the rotation θ_{3z} is given by Equation 2.80.

The final transformation is from joint 4 to joint 1. This is equivalent to transforming from the last joint to the end effector in an open-loop robot transformation. Typically six parameters are needed for this transformation. As the joint 4 displacement can be obtained from the constraint equations, this removes one independent parameter. To transform from joint 4 to joint 1, therefore, five parameters are needed. From the analysis shown above, to completely define the motion of a four-bar mechanism starting from the world coordinate system, 19 parameters are needed.

2.8.4.2 A 5R Mechanism Another example is shown in Figure 2.16. This is a 5R closed-loop mechanism with two driving devices located on joints 1 and 5. Similarly, four parameters are needed to transform from the world coordinate to joint 1 and from joint 1 to joint 2. To fix the orientational axes of joints 3, 4, and 5, three parameters are also needed on each transformation. To transform from joint 5 to joint 1, according to Equation 2.1, six parameters are needed. This is because joint 5 is also a driving device. Therefore, to define a 5R mechanism starting from the world coordinate system, 23 parameters are needed.

Based on the discussions above, the number of parameters to transform from one revolute joint to its next revolute joint is four. However, when the joint used for the transformation is not a driving device, the number of parameters reduces to three. The 5R mechanism needs four more parameters than the 4R mechanism.

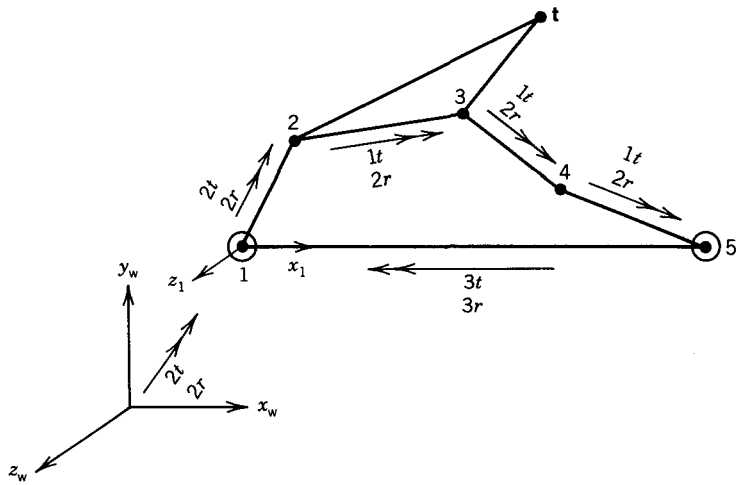


Figure 2.16. A 5R mechanism.

2.8.4.3 An RRRPR Mechanism To find the number of parameters required to describe a prismatic joint in a closed-loop mechanism, an example of an RRRPR mechanism is given. This mechanism is illustrated in Figure 2.17. Again, the driving devices are assumed to be on joints 1 and 5. To transform from the world coordinate system to joint 3 in Figure 2.17, the same parameters as those shown in the 5R mechanism can be used. To transform from joint 3 to

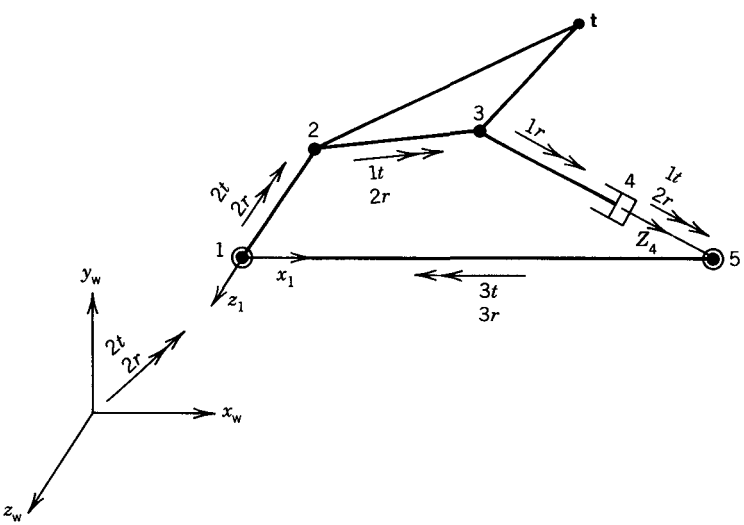


Figure 2.17. A closed-loop robot with an RRRPR mechanism.

joint 4, one can have the following two parameters:

$$\mathbf{R}(z, \theta_{3z})\mathbf{R}(y, \theta_{3y}) \tag{2.84}$$

However, since the value of θ_{3z} can be calculated by using the constraint equations, the number of parameters reduces to one. To transform from joint 4 to joint 5, one may have the following four parameters:

$$\mathbf{T}(z, l_{z4})\mathbf{T}(y, l_{y4})\mathbf{R}(z, \theta_{z4})\mathbf{R}(y, \theta_{y4}) \tag{2.85}$$

However, as the value of $\mathbf{T}(z, l_{z4})$ can be obtained by using the constraint equations, the number of parameters reduces to three. Similarly, as joint 5 is measurable, six parameters are needed to transform from joint 5 to joint 1. To define an RRRPR mechanism starting from the world coordinate system, therefore, 21 parameters are needed.

2.8.4.4 An RRSSR Mechanism As spherical joints are frequently used in 3D mechanisms, we will now consider a mechanism that includes this type of joint. Unlike a revolute joint, there is no single axis of rotation for a spherical joint. A spherical joint has to rotate at a unique point that is the center of the sphere. Based on this idea, three translation parameters are needed to define a spherical joint.

An RRSSR mechanism is shown in Figure 2.18. The driving devices are located on joints 1 and 5. To transform from the world coordinate system to joint 2, the same parameters as the 5R mechanism may be used.

As joints 3 and 4 are spherical joints, according to Denavit and Hartenberg [5], each spherical joint is equivalent to a combination of three revolute joints whose axes are mutually perpendicular at a common point of intersection. Figure 2.19 shows the sequence transformations between these two joints. Since the last

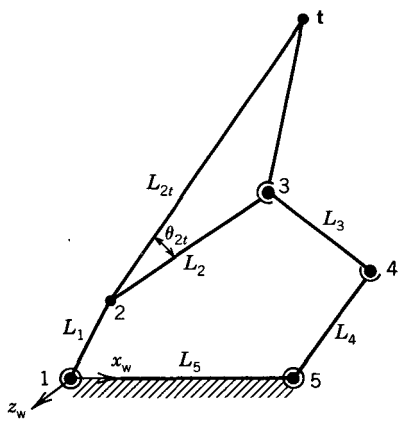


Figure 2.18. A closed-loop robot with an RRSSR mechanism.

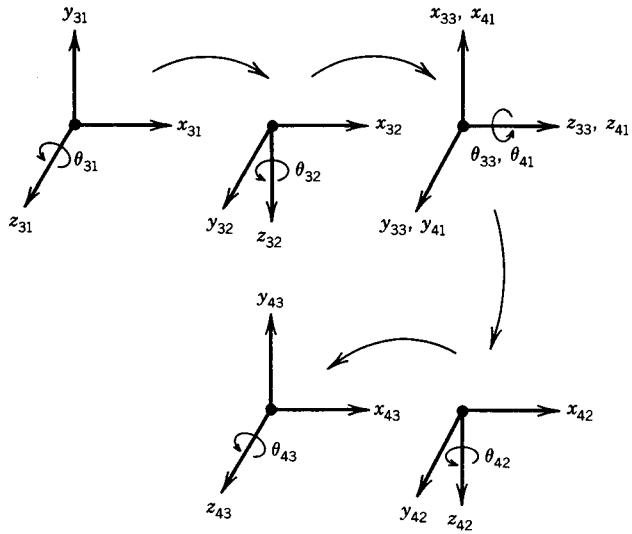


Figure 2.19. Coordinate transformation between two spherical joints.

coordinate system of joint 3 (X_{33}, Y_{33}, Z_{33}) is the same as the first coordinate system of joint 4 (X_{41}, Y_{41}, Z_{41}), there are only five dependent variables for each pair of spherical joints. In a closed-loop constraint, the maximum number of dependent variables is six. Therefore, the RRSSR mechanism, similar to the 5R mechanism, has two degrees of freedom.

Figure 2.20 shows the transformation from joint 2 to the spherical joint 3. The first coordinate system of joint 3 (X_{31}, Y_{31}, Z_{31}) can be flexibly adjusted such that each axis can be parallel with that of joint 2. Therefore, the following three

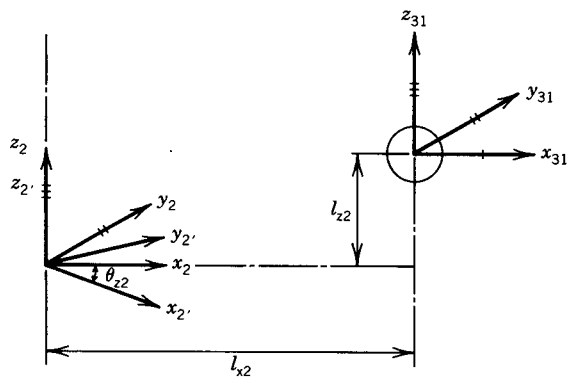


Figure 2.20. Transformation from a revolute joint to a spherical joint.

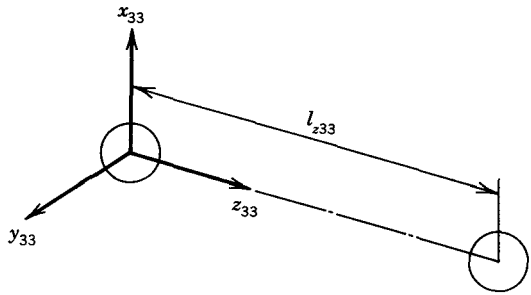


Figure 2.21. Transformation between two spherical joints.

parameters can be used to define the transformation:

$$\mathbf{T}(x, l_{2x})\mathbf{R}(z, \theta_{2z})\mathbf{T}(z, l_{2z}) \tag{2.86}$$

Again, since joint 2 is not a driving device, the second parameter can be determined elsewhere. To transform from joint 3 to joint 4, the third coordinate system of joint 3 (X_{33}, Y_{33}, Z_{33}) can be adjusted toward the center of joint 4. As shown in Figure 2.21, only one parameter is needed to make the transformation:

$$\mathbf{T}(z, l_{33z}) \tag{2.87}$$

where l_z is the distance between these two joints, and 33 is the third coordinate system of joint 3.

Figure 2.22 shows the transformation from joint 4 to joint 5. The third coordinate system of joint 4 can be flexibly adjusted such that each axis is parallel

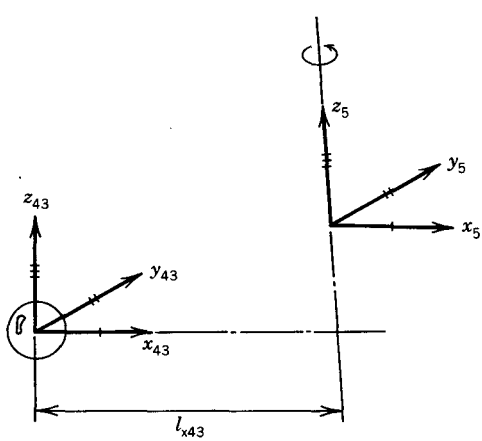


Figure 2.22. Transformation from a spherical joint to a revolute joint.

with joint 5. Therefore, only one parameter is needed:

$$\mathbf{T}(x, l_{43x}) \tag{2.88}$$

To transform from joint 5 to joint 1, six parameters are needed. Therefore, an RRSSR mechanism requires 18 parameters to be complete.

Clearly, the determination of the number of parameters required for completeness is more difficult for a closed-loop robot than a standard open-chain mechanism. The examples given above have been included to illustrate an approach to determining the proper number of kinematic model parameters for a manipulator with closed-loop chains.

2.8.5 Joints Comprised of Higher Pairs

Although most manipulators are designed with the intention that all of the joints will be either revolute or prismatic, it is physically impossible to construct a joint that will perfectly generate this type of motion. For example, most "prismatic" joints consist of a carriage constrained to move along a bar. Since the bar will be subject to some slight curvature or irregularities along its surface, the generated motion will not be purely prismatic. This phenomenon has been recognized in the area of calibration of coordinate measuring machines. In his report on software error compensation, Zhang et al. [11] reported the use of a rigid body model with six degrees of freedom per axis for a coordinate measuring machine to allow for imperfections in the machine axis. Figure 2.23 is an illustration of a joint that is intended to be prismatic but is subject to some error. As illustrated in the figure, the curvature in the bar causes the carriage to deviate from the desired path. This deviation induces an error in orientation as well as

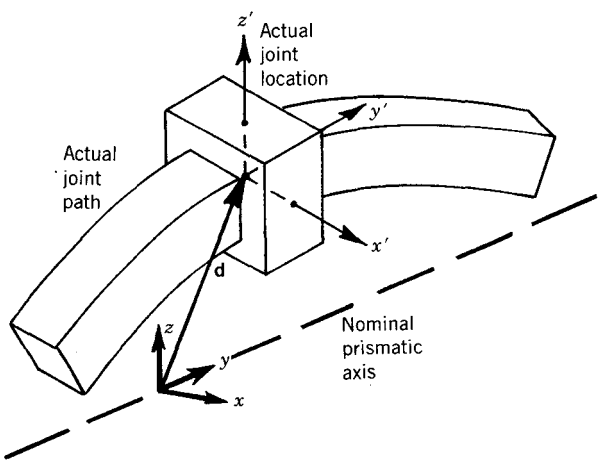


Figure 2.23. Higher order prismatic joint.

position. Since a prismatic joint will allow for translation only along a straight line and does not allow for a change in orientation during a displacement, these errors cannot be accounted for with a simple prismatic joint model. The axis of the nominal prismatic joint may be varied so as to minimize the effect of these errors over a given range, but the errors may not be eliminated. To account for these types of errors, the joint model may be modified so as to represent higher pair motion. Since most robots have joints that approximate lower pairs, the modeling process can be simplified. For example, we will consider the joint illustrated in Figure 2.23. The predominant motion of the carriage is along the bar. This implies that we may model the total motion as the combination of a prismatic displacement and a small additional motion to correct for the error. This may be expressed as

$$\mathbf{p}' = \delta\mathbf{S}\mathbf{S}\mathbf{T}_o^A\mathbf{p} \tag{2.89}$$

where S represents the motion of a prismatic joint and δS may be expressed

$$\delta\mathbf{S} = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.90}$$

where d_x, d_y, and d_z represent small displacements in the indicated directions and δ_x, δ_y, and δ_z represent small rotations about the coordinate axes. We shall refer to δS as the correction matrix.

Given this formulation, the process of calibration becomes one of determining the correct values of the correction matrix for each displacement of the joint. There are several important points to note about this process. First, the values in the correction matrix will depend on the orientation of the prismatic axis. Since the correction matrix will account for errors about the prismatic axis, the orientation of the prismatic axis may be specified to be some nominal value and omitted from the calibration procedure. If both the correction matrix and the prismatic axis are included in the calibration, their dependence on each other may result in numerical difficulties.

A second point to note is that the values in the correction matrix will be different for every possible displacement of the carriage. Since the shape of the bar is constant, however, a given displacement will repeatably result in a given correction matrix. In other words, the correction matrix is a time-indpendent function of the joint displacement. This implies that while the joint is a higher pair, it still has only one degree of freedom. To simplify the calibration process, it is desirable to choose a functional form for the terms in the correction matrix. Since the joint is intended to be prismatic, it is usually acceptable to assume that any term, δ_i, in the correction matrix will be a smooth, continuous, slowly varying

function of the joint displacement. The number of parameters to be identified, therefore, depends on the particular function chosen for each element of the correction matrix. For example if each term is modeled with a quadratic function such as

$$\delta_i = K_1 s^2 + K_2 s + K_3 \tag{2.91}$$

then each correction matrix will contain 18 parameters to be determined. The particular form of the function will be highly dependent on the expected level of deviation from true prismatic or revolute motion. It should be noted that if irregularities exist in the axis, the functions in the correction matrix may not be continuous. This may result when two bars forming the axis are not joined evenly and a "jump" results at the interface.

Since the number of parameters is dependent on the function chosen for the terms in the correction matrix, the concepts of completeness and equivalence do not apply for this type of model. For example, one manipulator may require that all terms in the correction matrix be approximated with harmonic functions whereas another may require only a few terms consistent with deflection in a given direction. Although both may be adequate for their specific situation, they would not have the same number of parameters and they would not be equivalent.

2.9 CONCLUSION

In this chapter, we have presented a number of issues that relate to the development of a suitable kinematic model for manipulator calibration. The concepts of completeness, proportionality, and equivalence have been introduced and demonstrated for several types of models and a number of robot geometries. Although these concepts can assist in the creation of a highly functional kinematic model with no redundant or dependent parameters, there is no guarantee that such a model is always the most appropriate for a given problem. Ultimately, the choice of the most appropriate level of model complexity will be a function of the robot construction, desired precision, and intended use. Typically, the goal of any modeling effort is to construct the simplest possible model that accurately reflects the phenomena of interest. For some robot tasks, the "best" model may be a simple level 1 model that ignores possible variations in the kinematic structure of the manipulator. If, for example, the manipulator is constructed to high tolerances and little variation in geometry is expected, the added complexity of a level 2 model would be of little benefit. On the other hand, if the manipulator under study has links that are subject to significant structural deflections, a complete level 2 model may not provide sufficient accuracy to meet the application requirements. In this case, a sophisticated level 3 model may be appropriate in spite of the significant increase in complexity. It is our belief that the material in this chapter will, then combined with experience and insight into the require-

ments of a specific situation, address the modeling requirements of most manipulators and intended applications.

REFERENCES

[1] H. Asada and J.-J. Slotine. *Robot Analysis and Control.* John Wiley & Sons, New York, 1986.

[2] J. Chen and L. M. Chao. Positioning error analysis for robot manipulators with all rotary joints. In *IEEE International Conference on Robotics and Automation,* pp. 1011–1016, April 1986.

[3] J. Chen and Y. F. Chen. Estimation of coordinate measuring machine error parameters. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation,* pp. 196–201, April 1987.

[4] John J. Craig. *Introduction to Robotics Mechanics and Control.* Addison Wesley, Reading, Massachusetts, 1986.

[5] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Transactions of ASME—Journal of Applied Mechanics,* 22(2): 215–221, June 1955.

[6] Morris R. Driels and Uday S. Pathre. Generalized joint model for robot manipulator kinematic calibration and compensation. *Journal of Robotic Systems,* 4(1):66–114, 1987.

[7] Morris R. Driels and Uday S. Pathre. Robot manipulator kinematic compensation using a generalized jacobian formulation. *Journal of Robotic Systems,* 4(2):259–280, 1987.

[8] L. J. Everett, Morris Driels, and B. W. Mooring. Kinematic modelling for robot calibration. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation,* pp. 183–189, April 1987.

[9] L. J. Everett and C. Y. Lin. Kinematic calibration of manipulators with closed loop actuated joints. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation,* pp. 792–797, Philadelphia, April 1986.

[10] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Vision, and Sensing.* McGraw-Hill, New York, 1987.

[11] G. Zhang, R. Veale, T. Chalton, B. Borchardt, and R. Hocken. Error compensation of coordinate measuring machines. *Annals of the CIRP,* 34(1):445–448, 1985.

[12] S. Hayati and M. Mirmirani. A software for robot geometry parameter estimation. SME Paper # MS84–1052, presented at Robots West Conference, Anaheim, CA, November, 1984.

[13] Samad A. Hayati. Robot arm geometric link parameter estimation. In *Proceedings of the 22nd IEEE Conference on Decision and Control,* pp. 1477–1483, December 1983.

[14] J. M. Hollerbach. A survey of kinematic calibration. In *The Robotics Review 1988.* MIT Press, Cambridge, MA, 1988.

[15] J. M. Hollerbach and D. J. Bennett. Automatic kinematic calibration using a motion tracking system. In *Modeling and Control of Robotic Manipulators and Manufacturing Processes, DSC*—Vol. 6, ASME Winter Annual Meeting, pp. 93–100, Boston, MA, December 1987.

[16] J. M. Hollerbach and D. J. Bennett. Automatic kinematic calibration using a motion tracking system. In *Robotics Research: The Fourth International Symposium*, pp. 191–198. MIT Press, Cambridge, MA, 1988.

[17] Tsing-Wong Hsu and Louis J. Everett. Identification of the kinematic parameters of a robot manipulator for positional accuracy improvement. In *Proceedings of 1985 ASME Computers In Engineering Conference and Exhibition*, pp. 263–267, 1985.

[18] R. Ibarra and N. D. Perreira. Determination of linkage parameter and pair variable errors in open chain kinematic linkages using a minimal set of pose measurement data. *Journal of Mechanisms, Transmissions, and Automation in Design*, 159–166, June 1986.

[19] Robert P. Judd and Al. B. Knasinski. A technique to calibrate industrial robots with experimental verification. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, pp. 351–357, April 1987.

[20] B. W. Mooring. The effect of joint axis misalignment on robot positioning accuracy. In *Proceedings of the 1983 ASME Computers in Engineering Conference*, pp. 151–155, August 1983.

[21] B. W. Mooring and G. R. Tang. An improved method for identifying the kinematic parameters in a six axis robot. In *Proceedings of the 1984 ASME Computers in Engineering Conference*, pp. 79–84, August 1984.

[22] Richard P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. The MIT Press, Cambridge, MA, 1981.

[23] D. Payannet, M. J. Aldon, and A. Liegeois. Identification and compensation of mechanical errors for industrial robots. In *Proceedings of the 15th International Symposium on Industrial Robots*, pp. 857–864, Tokyo, 1985.

[24] G. V. Puskorius and L. A. Feldkamp. Global calibration of a robot/vision system. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, pp. 190–195, April 1987.

[25] Henry W. Stone. *Kinematic Modeling, Identification, and Control of Robotic Manipulators*. Kluwer Academic Publishers, Boston, 1987.

[26] Henry W. Stone and Arthur C. Sanderson. A prototype arm signature identification system. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, pp. 175–182, April 1987.

[27] Henry W. Stone, Arthur C. Sanderson, and Charles P. Neuman. Arm signature identification. In *Proceedings of 1986 IEEE International Conference on Robotics and Automation*, pp. 41–48, April 1986.

[28] K. Sugimoto and T. Okata. Compensation of positioning errors caused by geometric deviations in robot system. In *Robotics Research: The Second International Symposium*, pp. 231–236. MIT Press, Cambridge, MA, 1985.

[29] C. H. Suh and C. W. Radcliffe. *Kinematics and Mechanism Design*. John Wiley & Sons, New York, 1978.

[30] Ramesh N. Vaishnav and Edward B. Magrab. A general procedure to evaluate robot positioning errors. *The International Journal of Robotics Research*, 6(1):59–74, 1987.

[31] W. K. Veitschegger and Chi-Haur Wu. A method for calibrating and compensating robot kinematic errors. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, pp. 39–44, April 1987.

[32] W. K. Veitschegger and Chi-Haur Wu. Robot accuracy analysis based on kinematics. *IEEE Journal of Robotics and Automation*, RA-2(3):171–179, September 1986.

[33] D. E. Whitney, C. A. Lozinski, and J. M. Rourke. Industrial robot forward calibration method and results. *Journal of Dynamic Systems, Measurement, and Control*, 108(1):1–8, March 1986.

[34] Chi-Haur Wu. A kinematic CAD tool for the design and control of a robot manipulator. *International Journal of Robotics Research*, 3(1):58–67, 1984.

[35] Chi-Haur Wu. The kinematic error model for the design of robot manipulators. In *Proceedings of the Americal Control Conference*, pp. 497–502, San Francisco, June 1983.

[36] H. Zhen. Error analysis of robot manipulators and error transmission functions. In *Proceedings of the 15th International Symposium on Industrial Robots*, pp. 873–878, Tokyo, 1985.

CHAPTER 3

MEASUREMENT TECHNIQUES FOR MANIPULATOR CALIBRATION

As described in the previous chapter, the purpose of a kinematic model is to relate the manipulator joint displacements to the pose of the end effector. This model will contain a set of coefficients that may describe the geometry of a particular robot or some nongeometric aspects of the robot motion. Calibration is simply the process of determining the set of parameters in the model that best describes the specific robot under study. Before the calibration can be completed, some set of measurements must be made that determines the actual position or orientation of some portion of the robot for a given set of joint displacements.

This chapter will describe the basics of measurement for manipulator calibration. The goal of the measurement process is to accurately determine either the end effector pose or some subset of the pose for a set of robot joint displacements. Typically, the measurement process consists of moving the end effector to some location in the workspace and recording the joint displacements. Next, the measurement system is used to accurately determine some portion of the pose. As described in the previous chapter, the robot model may be expressed as

$$\mathbf{T} = \prod_{i=1}^{N} \mathbf{A}_i \tag{3.1}$$

where $\mathbf{T}$ is a homogeneous transformation representing the pose of the end effector in the base coordinate system as predicted by the model and N is the number of joints in the robot structure. The actual pose, as determined by the measurement system, may be expressed as $\mathbf{T}_m$ and, in general, will not be exactly equal to the pose predicted by the manipulator model. In addition, $\mathbf{T}_m$ may not be completely known if the measurement system is not capable of determining a complete pose. For each pose, the known elements of $\mathbf{T}_m$ may be compared to the elements predicted by the model and expressed in $\mathbf{T}$. This will generate from

one to six independent equations relating the variations in the model parameters to the difference between the actual and predicted components of the pose. The solution of these equations for the optimal set of model coefficients is the topic of Chapter 4.

As stated earlier, the goal of the measurement step is to accurately determine either the complete end effector pose or some subset of the pose for a particular set of robot joint angles. The result of the measurement process, therefore, will be a data set that contains the joint displacements and some portion of the end effector pose for a number of robot configurations. To describe the various techniques available for acquiring the required data, this chapter is divided into three sections. Section 3.1 is a review of the transducers that are most commonly used for measuring joint displacement. Typically, the transducers that are built into the robot are used to determine the joint displacements. Since these will be the devices that are used during the robot operation, little benefit will be derived by using more precise devices during calibration. It is important, however, to understand the principles of operation and the limitations of these devices because these factors have a direct bearing on both the modeling and identification steps. Readers who are already familiar with the fundamentals of potentiometers, encoders, resolvers, and similar devices may wish to skip this section. Section 3.2 is a review of the devices used for pose measurement. In some cases, complete systems are described and in other cases, the principles of system components are discussed. The purpose of this section is to acquaint the reader with the basics of measurement technologies used in manipulator calibration. For example, there are discussions on the fundamentals of laser interferometers, coordinate measuring machines, theodolites, and a number of other devices. Those who have experience with these devices may wish to skip this section as well. In the final section, Section 3.3, measuring methodologies of robot calibration are presented. The goal here is to describe how various investigators have used the measurement hardware described above to acquire the data sets that are necessary for calibration. Measurement approaches are classed by the amount of pose information given by a single measurement. The relationship between measured quantities and end effector pose is given and references to those who have reported using the approach are specified.

As will become clear in following discussions, there is no "best" measurement system for robot calibration. A system must be chosen that provides the desired level of accuracy while meeting constraints of cost, size, and ease of use. The varying size and geometry of robots together with significant differences in their working environments lead to a wide variety of measurement approaches. The purpose of this chapter is to provide the basic information necessary to choose a suitable measurement system for a given robot application.

3.1 JOINT DISPLACEMENT TRANSDUCERS

Although robots are commercially available in a wide variety of sizes and geometries, the vast majority of designs rely on a fairly small set of joint displace-

ment transducers. This section begins with a description of potentiometers. Although inexpensive and reliable, these devices have serious shortcomings when used as joint displacement transducers for robots. The most popular devices by far are encoders and resolvers. These devices are described in the following paragraphs. The section concludes with a brief discussion of some of the less common joint displacement transducers. The purpose of this section is to give a brief introduction to this technology. Interested readers are referred to texts such as those by Doebelin [1] or Klafter, Chmielewski, and Negin [2] for more information.

3.1.1 Potentiometers

The simplest device available for measuring displacement is the potentiometer or "pot." A pot consists of a resistive element and a wiper that contacts the resistive element. Figure 3.1 is a schematic of a pot for measuring linear motion. As shown in the figure, the resistive element is a coil of resistive wire wrapped around a nonconducting base. For illustration, we will assume that the total resistance of the wire wrapped around the base is R. The base is affixed to the case of the pot, which is usually mounted to some portion of the robot frame. A voltage, V, is applied across the resistive winding and the wiper contacts the winding so that the voltage sensed by the wiper is proportional to the position of the wiper contact point. For example, if the position of the wiper is given by x as shown in Figure 3.1, the voltage sensed by the wiper V_w, will be

$$V_w = \frac{x}{L} V \tag{3.2}$$

where L is the length of the winding. If the wiper is attached to the moving portion of the robot joint, then the wiper voltage, V_w, will vary in proportion to the joint

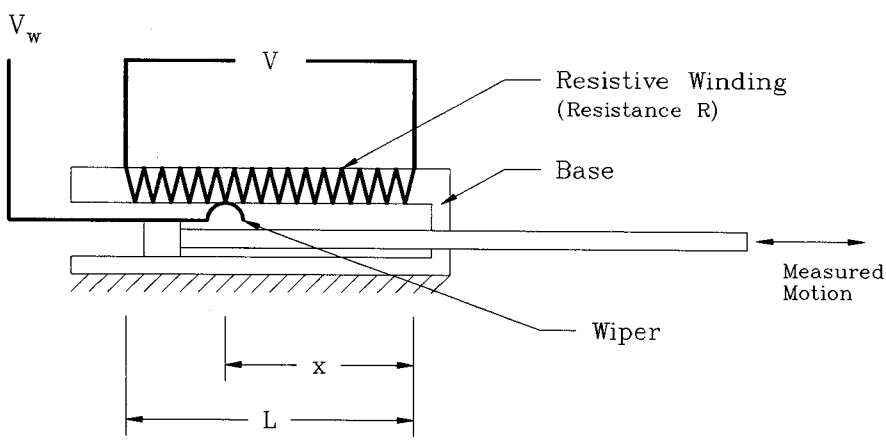


Figure 3.1. Schematic of a potentiometer.

displacement. This device may be reconfigured to measure rotary motion by attaching the wiper to a rotating shaft and forming the resistive element into an arc that the wiper may continuously contact.

There are several factors that limit the use of potentiometers for measuring joint displacement. The primary factor relates to the combination of range and "resolution" of the device. To illustrate this point, we will consider a revolute joint. If we assume that the joint rotates through 300°, it is not unusual to expect that the position of the joint be measured with an accuracy on the order of 0.001°. This implies that we must be able to discriminate at least 300,000 different voltage levels in the single rotation of the pot. If the pot is constructed with a wire wrapped around a base as described above, the wiper will be sliding in a direction that is perpendicular to the axis of the wire and, hence, will continually be moving from one coil to the next. This will result in discrete changes in voltage rather than a continuous variation. Typically, wirewound pots are not available with the required resolution. Pots are available, however, with a thin resistive film that replaces the wire winding. While these pots theoretically have an infinite resolution, effects such as electrical noise between the wiper and the film as well as nonlinearities in the resistance of the film significantly limit the variation in voltage that can practically be determined. Simply put, a pot does not provide the required precision when attached directly to the robot joint. A seemingly obvious solution to this problem would be to place a gear train between the pot and the joint so that the motion of the pot wiper is much larger than the motion of the joint. In this case, the shaft of the pot must move through a number of degrees for each degree of joint motion. This implies that the pot must be designed to operate through multiple rotations. While pots are commercially available with spiral windings that allow up to 20 rotations, they still do not provide the effective resolution that is necessary for a robot joint. A similar problem exists for linear pots. If a linear axis is to move over a distance of 3 feet, a pot directly connected to the axis must be 3 feet long. Clearly, this would create significant design problems. If the pot motion is decreased through a lever or gearing system, then effective resolution again becomes a problem.

Other problems that are related to the use of potentiometers for measurement of robot joint displacment involve the analog nature of the devices. Since the displacement signal is an analog voltage, any variation in this level will be perceived as joint motion. For example, if electrical noise in the environment is propagated to the wires carrying the wiper voltage, the controller will sense an erroneous displacement. Also, a similar problem exists if the reference voltage V_r is slightly varied. These and the problems described above have limited the usefulness of pots as joint displacement transducers to robots with low levels of precision that would not typically be candidates for calibration.

3.1.2 Encoders

An encoder is a device that may be used to measure linear or angular displacement. The encoder has proven to be extremely popular because it produces a digital signal that is easily interfaced to a computer. There are three distinct

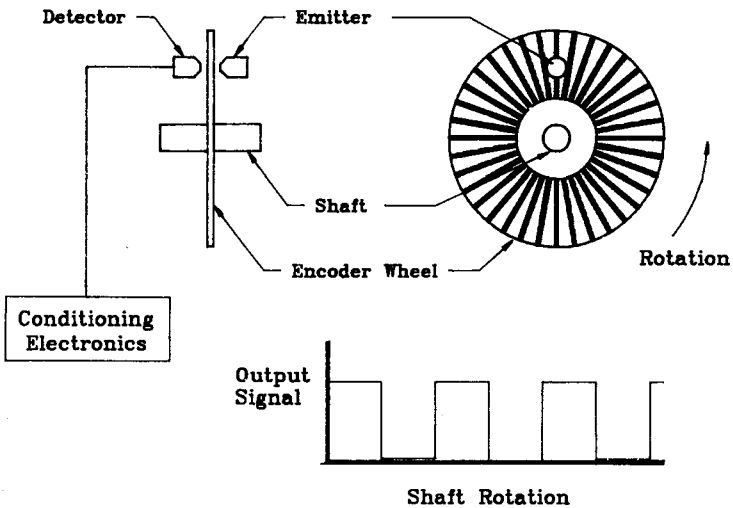


Figure 3.2. A simple encoder.

classes of encoders; tachometer, incremental, and absolute. Figure 3.2 is an illustration of the tachometer class of encoder. This device consists of a clear glass or plastic wheel with some number of opaque lines marked radially on the wheel. On one side of the wheel is a light source that is directed so the light will shine through the wheel. On the opposite side, a photo detector senses the light coming through the wheel. As the shaft rotates, the lines on the wheel cause the light to be alternately transmitted or blocked. The electronics associated with the photo detector produce a square wave such that the signal is at one level when the light is transmitted and at a different level when the light is blocked. The rotation of the wheel may then be measured by simply counting the pulses. The device illustrated in Figure 3.2 is termed a tachometer encoder because it is typically used for determining the rotational speed of a shaft that rotates in only one direction. This limited use is because there is no way to sense a change in direction of rotation. One cannot infer the direction of rotation by simply observing the output waveform of the encoder.

The incremental encoder addresses the direction problem by incorporating two pairs of light sources and detectors as illustrated in Figure 3.3. The first emitter–detector pair, referred to as the A pair, produces a square wave in the same manner as the tachometer encoder. The second pair or the B pair is offset so as to produce a square wave that is 90° out of phase with the wave produced by the A pair. The waveforms in Figure 3.3 illustrate how the direction of motion may be inferred from this information. If the A signal goes from low to high and is then followed by a B transition from low to high, the rotation is in a specific direction. If, however, the A signal goes from low to high and is followed by a B transition form high to low, we know that the motion is in the opposite direction.

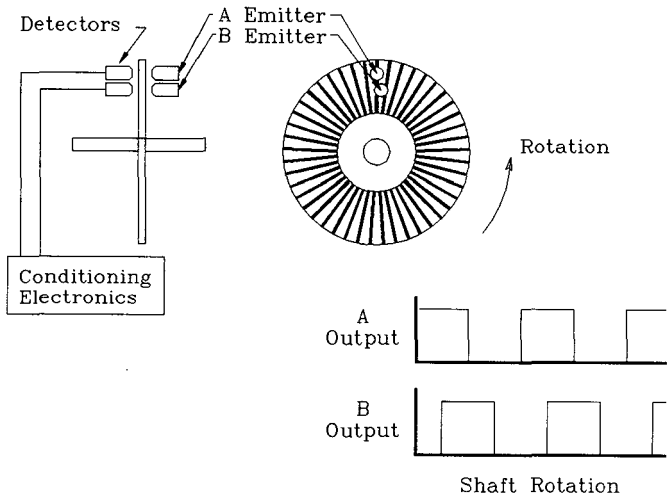


Figure 3.3. An incremental encoder.

An electronic circuit may be easily designed to sense these changes and keep a running count of the total number of pulses since the counting was initiated. The counter is designed to count up when the wheel rotates in one direction and to count down when the wheel rotates in the opposite direction.

The smallest amount of rotation that may be measured by an encoder is determined by the number of lines that are on the wheel. For example, if a wheel has a total of 360 lines and we choose to count the number of times that A signal goes high, we will not be able to sense changes in rotation of less than 1°. If, however, we choose to count each time the A line goes from low to high and each time the B line goes from low to high, we will get 720 counts per revolution. This would yield a resolution of 0.5°. Similarly, we may choose to count high to low as well as low to high transitions of both the A and B signals. This would give us a total of 1440 counts per revolution or a resolution of 0.25°. In summary, an encoder with n lines per revolution may be set up to provide n, $2n$, $4n$, counts per revolution.

One major difficulty with the incremental encoder is the initialization of the counter. When power is first applied to the device, the counter must be initialized to some value. This impiles that some external means of determining the original position of the encoder must be established. For example, the shaft may be rotated to a hard stop, which represents 0° of rotation. The counter may then be loaded with zero so that all subsequent motions will be referred to an accurate base. In many applications, however, the determination of the initial count is difficult and may lead to a significant error if not done to a level of accuracy that is consistent with the resolution of the encoder.

The problem of initialization is avoided by the absolute encoder. As illustrated in Figure 3.4, the absolute encoder has a number of tracks and one emitter–

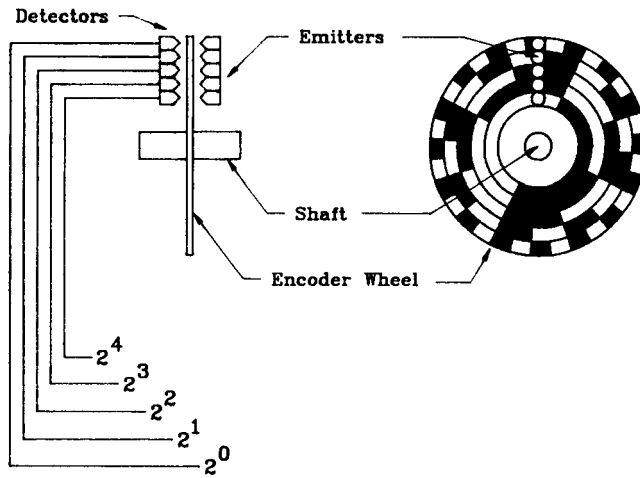


Figure 3.4. An absolute encoder.

detector pair on each track. The clear and opaque regions in each track are designed so that when all of the detectors are read, they produce a code that indicates the angular orientation of the wheel. For example, the wheel illustrated in Figure 3.4 has five tracks. If the wheel is in the position shown in the figure, the output from the emitter–detector pairs will be 01101. If this is treated as a binary number, it indicates that the wheel is in position 13. It is important to note that the output is immediate and does not depend on a running count. Also, the resolution of the device depends on the number of tracks that are on the wheel. For example, if the wheel has five tracks, only the binary numbers from 00000 to 11111 may be represented. This represents a range of only 32 numbers. If a resolution of 1000 steps per revolution were required, the encoder would require at least 10 tracks. Another problem associated with this type of encoder is that the output code begins to repeat itself after one revolution. If multiple rotations of the input shaft are desired, an external counter similar to that of the incremental encoder must be added.

While the devices described above measure rotary displacement, encoders of all three classes are available to measure linear motion. As illustrated in Figure 3.5, the clear wheel is replaced with a transparent strip. Lines are marked on the strip in patterns similar to those on the rotary units. Typically the strip is attached to a fixed base and the sensing head with the emitter–detector pair is attached to the moving body.

A typical data collection procedure for a manipulator calibration requires that the position of points on the end effector be accurately determined. If an encoder is to be used as part of this measurement, it must be a high-resolution device that is sensitive to the direction of the displacement being measured. These requirements tend to make the tachometer encoder and the absolute encoder

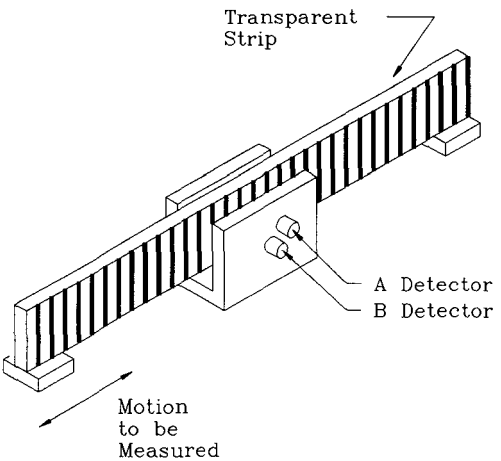


Figure 3.5. A linear encoder.

impractical for use in this application. The tachometer encoder cannot sense the direction changes that are necessary for this type of measurement. Most absolute encoders do not have the resolution required for the task. For example, it is not uncommon to require that angular measurements be made with a resolution of better than 5 arc seconds (0.0014°). This implies that the absolute encoder would require 259,200 divisions or 18 tracks to have the necessary resolution. The expense and size of such a device preclude its common use.

The advantages of the incremental encoder for this application are its digital output, resolution, and relatively low cost. As mentioned earlier, the waveform generated by the incremental encoder is easily interfaced to a digital computer. The problems with drift and noise that exist in analog transducers are minimized in this device. Low cost incremental encoders are available with up to 1000 lines per revolution. This offers a resolution of 0.36° for 1 revolution. Because the incremental encoder is not limited to a single revolution, precision gearing may be used so that the encoder will make a number of revolutions for each turn of the shaft being measured. Although this approach can significantly improve the resolution, effects from gear backlash and eccentricity may affect the accuracy of the total system. Encoders with higher resolution are available, but the cost increase is substantial as the resolution is increased.

3.1.3 Resolvers

A resolver is a device that may be used for measuring displacement of a revolute joint. The transducer is based on the concept of a rotating transformer that has two rotor windings and, typically, two stator windings. A schematic of this device is illustrated in Figure 3.6. To measure position, one of the stator windings is shorted and the other is excited with a constant amplitude ac signal. This

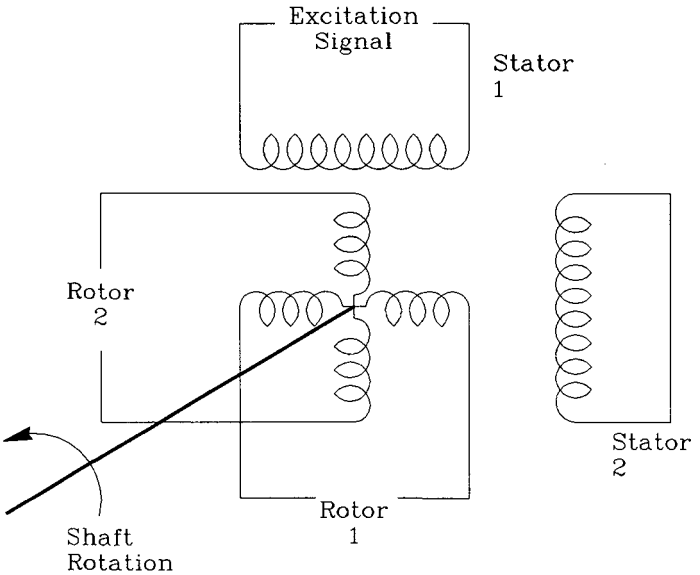


Figure 3.6. Schematic of a resolver.

excitation signal is typically either 60 or 400 Hz. The rotor coils are designed so that the excited stator coil induces a different voltage in each coil. If the amplitude of the excitation voltage is V and the excitation frequency is ω_E, the voltage in each rotor coil will be given by

$$V_{r1} = V \sin\theta \sin\omega_E t \tag{3.3}$$

$$V_{r2} = V \cos\theta \cos\omega_E t \tag{3.4}$$

where θ is the angle of the rotor shaft. Clearly, the amplitude of V_{r1} is $V \sin\theta$ and the amplitude if V_{r2} is $V \cos\theta$. The ratio of the amplitudes may be formed to give $\tan\theta$. Solid-state devices are available to transform the rotor voltages into a voltage that is proportional to the shaft displacement.

Resolvers have a number of advantages. The primary advantage is that the devices give an absolute reading of the joint displacement and do not require the initialization that an incremental encoder does. In other words, the joint position is known at power up and no "homing sequence" is required. Also, typical resolutions are better than encoders or pots and, therefore, the device may be attached directly to the joint axis. Inexpensive electronics packages are available that take the analog rotor voltage and a 12-bit digital signal corresponding to a desired shaft angle as the input. The output is a 12-bit digital signal representing the error between desired and actual shaft angle. The combination of the resolver and the resolver converter is a perfect match for the controller, which uses the position error to generate an updated desired shaft angle. In

addition to the advantage cited above, these devices are rugged and can be configured to measure velocity as well as displacement.

As with all transducers, resolvers have several disadvantages that tend to limit their use in robotics. The foremost is cost. Typically, a resolver with its associated electronics is significantly more expensive than an encoder package with comparable resolution. Other problems relate to electrical interference created by the carrier signal and the larger number of wires that must be run to the device.

3.1.4 Less Common Devices

Several devices other than the ones mentioned above have been used by commercial robots to measure joint displacement. One of the most interesting is a device marketed by Temposonics, Inc. using sonic wave propagation. The device, illustrated in Figure 3.7, consists of a magnetostrictive wire contained in a protective tube of nonferrous material. At one end of the tube, the wire is attached to an electronics package that can induce a current pulse through the wire. A ring magnet is located around the tube and is attached to the part of the robot whose displacement is to be measured. When the current pulse passes the magnet, the magnetostrictive action in the wire creates a stress pulse that propagates back to the electronics package at the speed of sound in the wire (approx. 110,000 in./sec). The time interval between the initiation of the current pulse and the arrival of the stress pulse at the electronics package is proportional to the displacement of the magnet. The speed of sound in the wire is such that the pulse takes about 9 μsec to travel 1 in. A 10-MHz counter may be used, therefore, to obtain a displacement with a resolution of approximately 0.010 in. This device was used to obtain joint freeback in the IBM 7565 robot. Although the resolution is somewhat low, the device has the advantages of being relatively rugged, it does not have to be initialized, and it is simple to implement in a joint design.

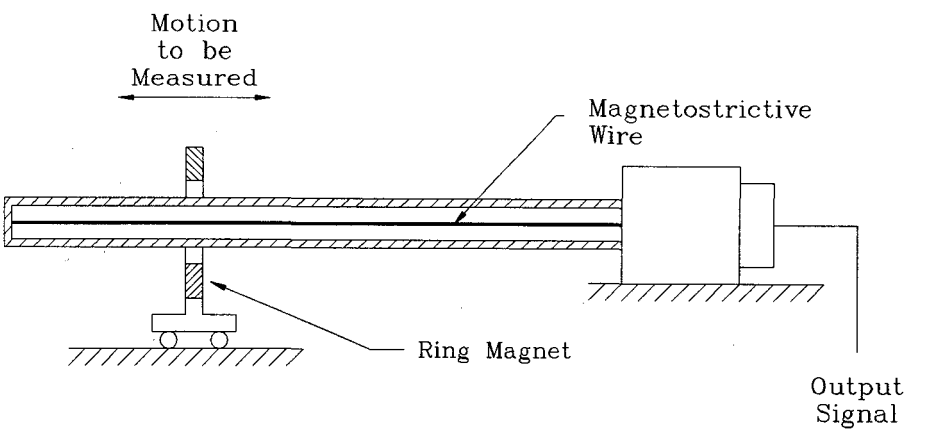


Figure 3.7. The Temposonics displacement transducer.

Another device that has been used on robots requiring extreme precision is the laser interferometer. Since this device has found wider application in pose measurement than as a joint feedback device, it is described in more detail in Section 3.2.2. Similarly, the LVDT has been used as a joint displacement transducer but it is more commonly used in the workspace. It is described in Section 3.2.6.

3.2 VARIOUS MEASUREMENT TECHNOLOGIES

A number of different techniques have been used to acquire the data necessary for manipulator calibration. Before discussing these approaches in detail, however, it is beneficial to review the basic measurement technologies on which these techniques are based. The following sections describe the fundamental principles of several measurement systems that are applicable to manipulator calibration. Although these have been the most popular for calibration, they represent only a small subset of the many measurement technologies that are currently available. Textbooks such as the one by Doebelin [1] provide a much more exhaustive treatment of this field.

3.2.1 Theodolites

A theodolite is simply a telescope that has been instrumented so that the line of sight is very precisely known. The line of sight is usually defined by two angles. The first is the angle between the line of sight and the horizontal plane; the second is the angle between an arbitrarily chosen horizontal line and the plane formed by the vertical axis and the line of sight. The text by Cooper [3] provides a detailed description of the construction and operation of modern theodolites. As described by Cooper [3] and illustrated in Figure 3.8, the theodolite consists of three parts; the base, the alidade, and the telescope. The base is usually mounted to a tripod or similar stand. The vertical axis is established by adjusting the base until it is very nearly horizontal as indicated by a set of bubble levels attached to the base. The alidade rotates about the vertical axis and is instrumented so that the rotation may be precisely known. Until recently, this rotation was measured manually by reading a vernier scale. Although measurements as precise as ± 0.5 arc second were possible, correct reading of the vernier was time consuming and subject to error. Fortunately, modern theodolites are equipped with digital displays that report the angles to ± 0.5 arc second. The digital output reduces the chance for error in making the readings and allows the device to be directly interfaced to a data acquisition system. The alidade has two vertical supports or standards which house bearings that form the second axis of rotation for the telescope. This axis is instrumented to the same level of precision as the vertical axis.

To use the theodolite, the unit is set up in a fixed position and the vertical axis is established by leveling the base. The operator then sights through the

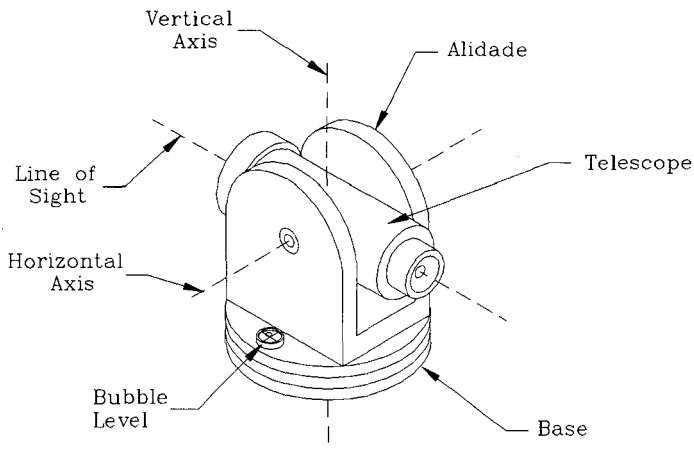


Figure 3.8. A theodolite.

telescope until the target comes into view. The telescope is focused on the target and aligned until the target point is centered on the cross-hairs in the telescope. The vertical and horizontal angles that establish the line of sight are then read. It is important to note that the theodolite does not provide a distance reading.

3.2.2 Laser Interferometers

The laser interferometer uses light interference principles to precisely measure the linear displacement or velocity of a body. A simple interferometer is illustrated in Figure 3.9. A laser produces a beam of coherent, monochromatic light that is passed through a beam splitter. Part of the beam is reflected toward a fixed mirror and the other part of the beam is transmitted through the beam splitter toward a moving mirror. The light from both the fixed and moving mirrors is reflected back toward the beam splitter, which is designed to recombine the beams as illustrated in Figure 3.9. As the moving mirror is displaced, the recombined beams will constructively and destructively interfere. As a result, the photodetector will sense an alternating intensity. A complete cycle of interference (light to dark) represents a mirror displacement of one-half a wavelength of the light from the laser. Since visible light has a wavelength of approximately 2.56×10^{-5} in., the resolution is well within that required for calibration measurements.

The accuracy of the interferometer will be affected by any influences that alter the wavelength of light. When light travels through air, the wavelength is affected by the air pressure, humidity, and temperature. Since these effects are often small, they may be ignored in many cases. Commercial systems are available, however, that compensate for these effects.

Figure 3.10 illustrates a modern laser interferometer system [1]. The differ-

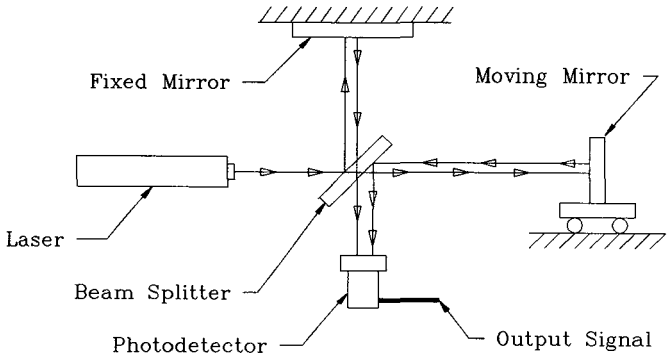


Figure 3.9. A basic laser interferometer.

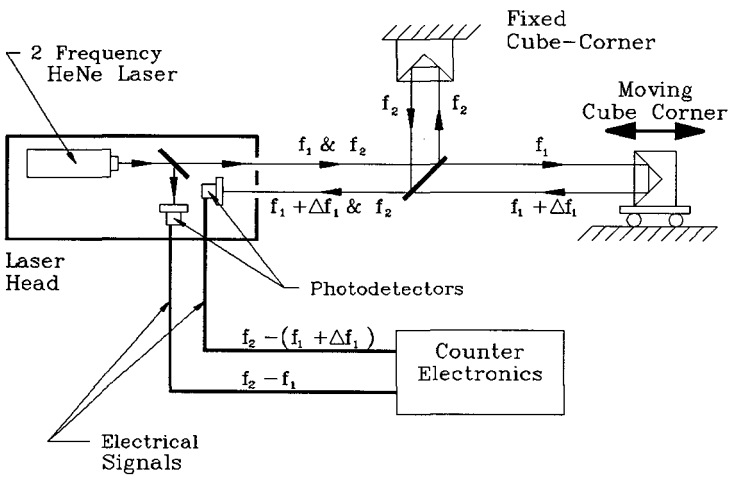


Figure 3.10. A modern laser interferometer. Redrawn with permission from Hewlett Packard, Palo Alto, CA.

ences between this system and the simple one described above improve the precision, range, portability, and ease of use. The system illustrated is also capable of making velocity measurements. The laser in this system is designed to produce light composed of two basic frequencies, in the range of 5×10^{14} Hz but separated by 2 MHz. In addition, the light at each frequency is polarized in a different direction. As with the simple system, the beam from the laser is directed through a beam splitter that transmits part and diverts part of the beam. The diverted beam or the "reference" beam is then passed through a set of polarizing filters to separate the two frequencies. The light of both frequencies is then directed to a photodetector. Since the two frequencies differ by 2 MHz, they will constructively and destructively interfere so that the photodetector will sense an intensity variation with a frequency of 2 MHz.

The light that was transmitted through the beam splitter is termed the "measurement" beam. The measurement beam is directed into a fixed interferometer. This device consists of a polarizing beam splitter and a cube corner. The polarizing beam splitter isolates the higher frequency and directs it toward the corner cube. The corner cube directs the light back toward the beam splitter, which is designed to point this beam back in the direction of the laser. The lower frequency light is passed to another corner cube that is attached to the object to be measured. This light is also reflected back toward the laser. Both reflected beams are recombined and directed through a set of polarizers and onto a photodetector. If the corner cube attached to the measured object is still, the two measurement beams will recombine to produce a varying intensity with a frequency of 2 MHz exactly as the reference beams. If, however, the corner cube is moving, its reflected beam will exhibit a doppler shift in frequency and the frequency of the intensity variation of the recombined measurement beams will vary. The frequency variation will be in direct proportion to the velocity of the corner cube. The outputs from both the reference and measurement photodetector are then directed to counters. The difference between the two counts is proportional to the displacement of the measured body.

There are two primary problems associated with the use of a laser interferometer for robot calibration. The first is the establishment of a reference position. As described above, an interferometer will accurately measure displacement from some point by counting the number of interferences in the reflected light. If the light beam is interrupted at any time, the measurement must be begun again. This implies that the robot end effector must be in a known position when the measurement is begun or some way of calibrating the laser must be developed so that the beam to the end effector is not affected.

The second problem with interferometers is cost. A typical commercial laser interferometer with the necessary optics will cost approximately $30,000. This cost does not include a system to point the beam at the target or an interferometer calibration system.

3.2.3 Coordinate Measuring Machines

A coordinate measuring machine (CMM) is basically a 3 DOF mechanism with three orthogonal prismatic axes. These devices are manufactured to ensure precise motion along the desired axes and they are instrumented to determine the joint displacement to a high degree of accuracy. A typical CMM is illustrated in Figure 3.11. Coordinate measuring machines are available in a wide range of sizes at various levels of precision. At the "low end" of size and performance, the axis of motion are formed with hardened shafts and linear bearings. Each joint is instrumented with either a linear encoder or a rotary encoder driven with a rack and pinion gear arrangement. Such devices typically have a resolution of 0.001 in. and a total accuracy on the order of 0.005 in. A typical low end CMM is illustrated in Figure 3.12. At the high end of size and performance, CMMs may

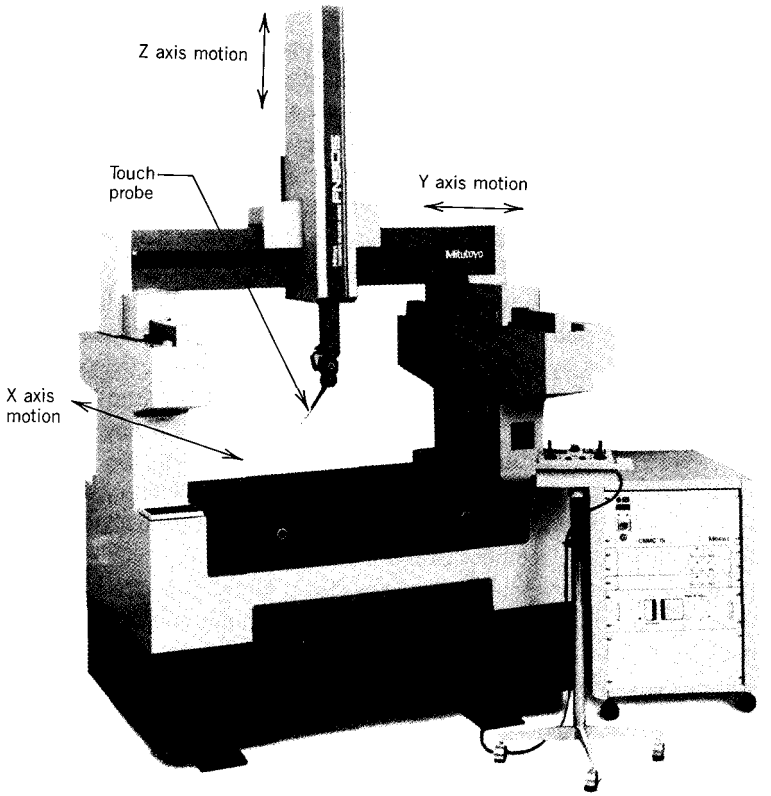


Figure 3.11. A typical coordinate measuring machine (CMM). Photograph courtesy of Mitutoyo/MTI Corporation, Paramus, NJ.

have highly precise axes that ride on air bearings, laser interferometers for determination of axis motion, and work volumes of over 1000 ft^3. The accuracies of these machines may be on the order of a few ten-thousandths of an inch.

Typically, coordinate measuring machines are designed for inspection of parts and assemblies. Most CMMs large enough for robot calibration are too expensive to be justified solely for that purpose. Some low end CMMs, such as illustrated in Figure 3.12, are suitable for calibration of small assembly robots like the PUMA 250 and 560.

3.2.4 Time of Flight Devices

A popular means of measuring the distance between two bodies is to have one body emit a signal that travels at a known rate. A sensor on the second body receives the signal and the time of travel between emitter and receiver is recorded. The time of flight together with the known speed of the signal will indicate the distance traversed. Currently, the most popular signal for this type of measure-

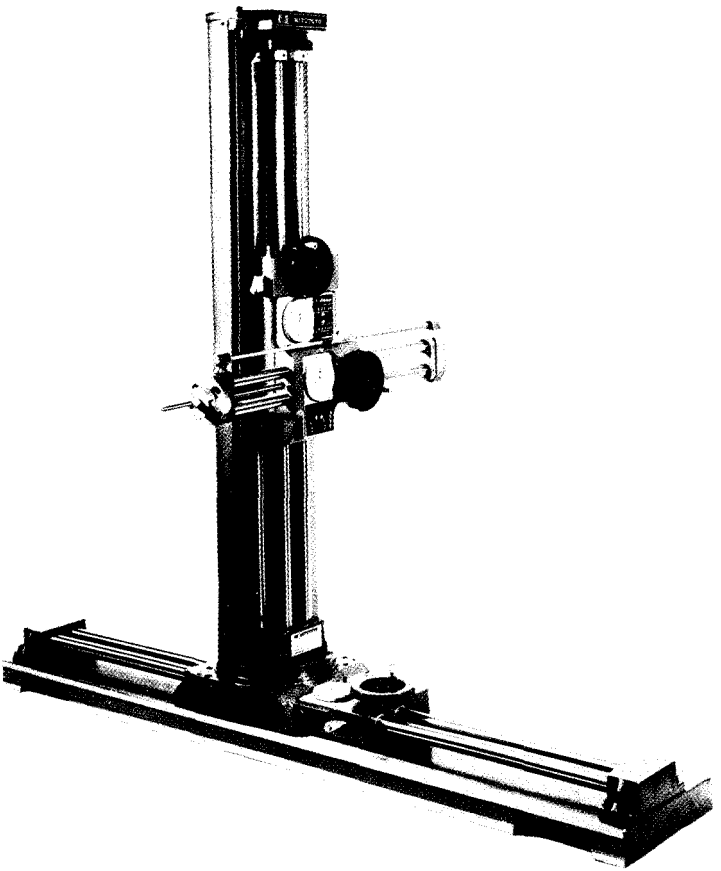


Figure 3.12. The Mitutoyo CX-D2 coordinate measuring machine. Photograph courtesy of Mitutoyo/MTI Corporation, Paramus, NJ.

ment is a sonic pulse. At first thought, it would seem desirable to use a pulse of light for time of flight measurement because the speed is well known and constant. Unfortunately, light travels much too quickly to make time of flight practical over the relatively small distances involved in robot calibration. For example, a light pulse would take only 5.083×10^{-8} sec to travel 50 feet. If we wanted to make this measurement with a resolution of 0.001 in., it would be necessary to measure time to within 8.47×10^{-14} sec. Clearly, this would not be at all practical for our application.

A popular approach to time of flight measurements is to emit a sound pulse in air and measure the time required for the pulse to traverse the distance between the emitter and receiver. The speed of sound in air at 70°F is approximately 1128 ft/sec. At this speed, a sound pulse would require about 7.38×10^{-8} sec to travel 0.001 in. This falls well within the resolution of commonly available electronic

timers. The main problem with this approach is the variation on the speed of sound in air. The speed of a sound pulse is a function of temperature, humidity, and convection currents in the air. For example, the speed of sound increases by 11 ft/sec if the temperature is raised from 70 to 80°F. Since all of these conditions are dynamic, any measurement system based on time of flight of a sound pulse in air must continually compensate for these effects. Another difficulty that can arise with this approach is the effect that obstacles in the measurement space can have on the sound pulse. An obstacle between the emitter and detector can significantly affect the perceived time of flight.

3.2.5 Camera-Type Devices

With the advent of low cost computer systems and the development of the charge coupled device (CCD) camera, vision systems have seen a sharp rise in popularity. A typical vision system consists of a lens and a light-sensitive array. Light from the image is focused on the array by the lens. The light-sensitive array consists of a large number of discrete cells that sense the frequency of the light that strikes them. These cells are referred to as picture elements or pixels. The frequency information from each pixel is then digitized and a number is assigned to the sensed frequency. For example, one cell may see a completely black input and return a value of 16 while another that sees a very light gray color may return a value of 2. Using this information from each cell, the image may be reconstructed and analyzed.

When used as a part of a measurement system, one of the primary considerations is the resolution of the array. For example, assume that an array consists of 512 rows and 512 columns. If the lens system is focussed so that a 5 ft by 5 ft workspace is projected on the array, each pixel would be averaging the light from a 0.117 by 0.117 in. square. Although image processing techniques are available to estimate the location of known shapes to tolerances smaller than that represented by one pixel, the utility of the vision system as a measurement tool is ultimately limited by the resolution. If a resolution of 0.002 in. is required for a given identification procedure and the vision system to be used has a 512 by 512 array, the measurement volume will be limited to no more than a few inches.

Another camera type measurement system is manufactured by Selspot. A similar device is also manufactured by Northern Digital, Inc. These systems consist of a camera with a photosensitive detector that is sensitive to light in the infrared range. The sensor is capable of accurately determining the location of the center of a spot of light that has been projected onto its surface. The detector is an analog rather than digital device so the resolution is theoretically infinite. The device is also highly linear, which enhances the accuracy of the total system.

These systems operate in the following manner. A number of infrared LEDs are attached to points to be measured. The system controller then switches each LED on in sequence and the light from the LED is projected through the lens onto the photodetector. The X–Y coordinates of the image on the detector are proportional to the X–Y location of the LED in the field of view. One LED

may be located in approximately 50 μsec so the system can record the trajectory of a number of LEDs. The Selspot that is currently on the market is advertised to have a repeatability of 0.005% of the measuring range and a nonlinearity of 0.1% of the measuring range.

3.2.6 Short-Range Devices

Several manipulator calibration procedures have been proposed that are based on the use of a set of fixtures. These fixtures are located at a number of positions in the workspace and are designed to establish a local coordinate frame at each location. The fixtures are accurately placed in the workspace so the relationship between each local coordinate system and the reference system is known. The accurate placement of the fixtures is not a trival task and many of the measurement approaches that have been used directly for robot calibration may be employed for determination of fixture location. This approach is advantageous if the global measurement system is costly or would create too many obstacles in the workspace. For example, a laser interferometer system might be used to very accurately determine the location of the fixtures. The laser system may then be removed for use elsewhere, thus reducing the total cost of the measurement system. Each local fixture is instrumented so that small variations within the local coordinate system may be accurately measured. This allows the use of short-range sensors that can provide accurate measurements at a significantly lower cost than a global sensor of equivalent accuracy. The following paragraphs describe the principle of operation of several short-range position sensors that have been used in fixtures for robot calibration.

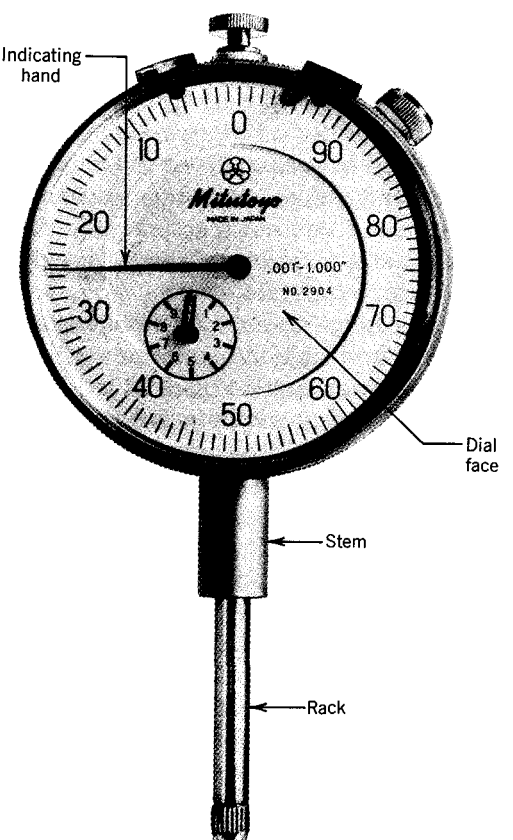
The most popular short range sensor for calibration has been the dial indicator. Dial indicators are available in two types: mechanical and digital. A mechanical dial indicator is illustrated in Figure 3.13a. This device consists of a spring loaded rack that may be depressed into the case. The amount of travel is recorded by the indicator arm, which is connected to the rack through a precision gear train. These devices are available in a large variety of strokes and may have resolutions of 0.0001 in. Although they are very inexpensive, mechanical dial indicators must be read manually and, thererfore, are not suited to an automated calibration procedure.

A digital dial indicator is illustrated in Figure 3.13b. In this device, a linear encoder is attached to the rack so that the displacement is measured electronically. These units are easily interfaced to computer systems that allow the automation of the measurement process. Digital indicators are available with a resolution of 0.0001 in. for as little as $250.

Another device for measuring small displacements is the linear-variable differential transformer or LVDT, which is illustrated in Figure 3.14. The LVDT consists of a set of coils and a core that is attached to the object that is being measured. The primary coil is excited with a sinusoidal voltage and the two identical secondary coils have induced in them a sinusoidal voltage of the same frequency. The amplitude of the voltage in each coil is a function of the position

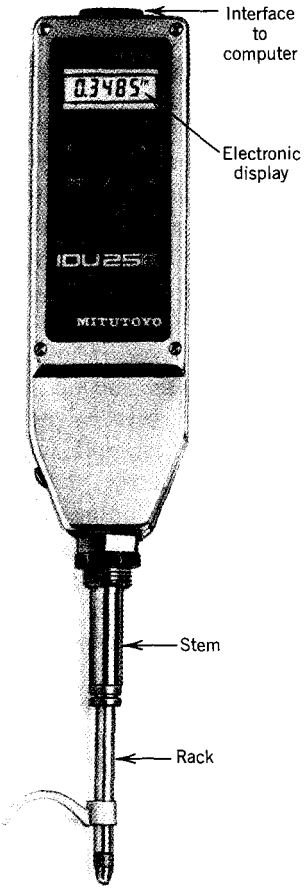
of the core that serves to enhance the coupling between the primary and secondary coils. When the secondary coils are arranged as shown in Figure 3.14 and the core is in the middle, an equal and opposite voltage is induced in each secondary coil. These cancel each other out and no voltage appears at the output. When the core is displaced, however, more voltage is induced in the secondary coil that is in the direction of the displacement. The result is an output voltage that is very nearly a linear function of the core position.

LVDTs are commercially available in a wide range of sensitivities and strokes. For example, standard units having strokes from ± 0.005 to over ± 3.00 in. are commonly available in sensitivities as high as 1.5 V/0.001 in. Since the coupling phenomenon is continuous, the resolution of the LVDT itself is infinitesimal. The overall measurement resolution, therefore, is a function of the quality of the electronics used to sense the variations in output voltage.



(a)

Figure 3.13. Dial indicators. Photograph courtesy of Mitutoyo/MTI Corporation, Paramus, NJ.



(*b*)

Figure 3.13 (*continued*). Photograph courtesy of Mitutoyo/MTI Corporation, Paramus, NJ.

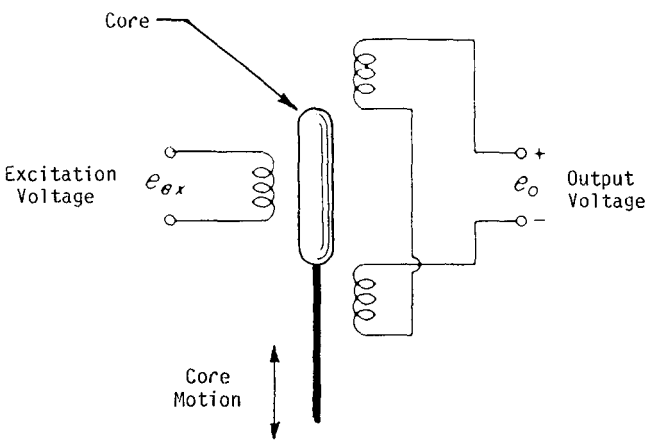


Figure 3.14. A linear-variable differential transformer (LVDT).

3.3 MEASURING METHODOLOGIES FOR ROBOT CALIBRATION

In the previous section, several basic measurement technologies were described. Some of the devices were complete measurement systems and others were components from which a system might be built. In this section, we will investigate how these measurement technologies may be applied specifically to the problem of data acquisition for manipulator calibration.

A number of different approaches to the measurement problem have been attempted. Rather than simply listing these various approaches, an attempt will be made to group them on the basis of the information provided by the measurement system. Every measurement system is designed to provide a specific set of information for each robot pose. For example, one system might provide the coordinates of a point on the end effector (three independent values) while another may yield the complete pose including position and orientation of the end effector (six independent values). In this work, we will classify each measurement system by the number of independent quantities determined for each robot pose. Although there will always be some gray areas and overlap, we feel that this classification will be a valid grouping of the various approaches.

3.3.1 Single Theodolite

The first useful group would be those measurement systems that provide two independent quantities per pose. The only known measurement system in this class would be a single theodolite. For each pose, the theodolite measures a vertical angle and a horizontal angle that establishes a line in space. If the theodolite has been pointed at a target on the end effector, it is known that the target lies somewhere along the line. The location of the point along the line, however, is completely unknown. Whitney, Lozinski, and Rourke [4] used a single theodolite measurement system to accomplish the calibration of a PUMA 560 manipulator. The theodolite is set up so that the target on the end effector may be viewed in as large a subset of the workspace as possible. The world coordinate system, XYZ_w, is established at the intersection of the theodolite axes. As illustrated in Figure 3.15, the theodolite is pointed at the target, which lies at the center of the tool coordinate system, XYZ_t. If the horizontal angle is given by α, the vertical angle by β, and the distance to the tool point by r, the transformation from XYZ_t to XYZ_w will be given by:

$$\begin{bmatrix} ? & ? & ? & r\cos(\beta)\cos(\alpha) \\ ? & ? & ? & r\cos(\beta)\sin(\alpha) \\ ? & ? & ? & r\sin(\beta) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

where the question mark represents elements of the transformation that are not determined by the measurement system. The variable r has been included in the

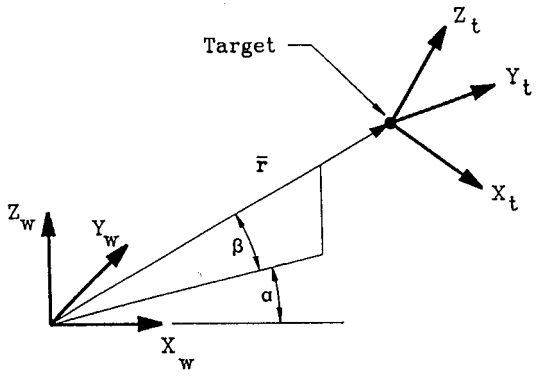


Figure 3.15. The single theodolite approach.

transformation but it is not determined by the theodolite. This implies that while the ratios of the elements in the last column are known, the absolute value of each is unknown. This has an interesting effect on the resulting parameter identification process. All of the unknown angle parameters may be identified directly. The length parameters, however, may be determined only as ratios with an arbitrary constant. This mathematical result may be interpreted physically by imagining the view through the telescope. As the robot is moved from one pose to the next, the observer would have no way of knowing if he were viewing a small robot at close range or a very large robot at long range. This lack of a basic length scale dictates that at least one additional measurement be taken. A minimum of two points on the manipulator that are a known distance apart should be viewed through the theodolite and the angles recorded. This will allow the length scale to be set and the absolute values of the length parameters to be determined.

The single theodolite approach has several advantages. Theodolites are common devices and may be purchased or rented at a reasonable cost. The target does not have to be elaborate. Whitney, Lonzinski, and Rourke [4] used a 0.3-mm-diameter sphere attached to a short post that was mounted on the last link of the robot. It should be noted that most theodolites are constructed to sight targets that are a very large distance from the instrument. The telescope may not be able to focus on targets closer than 10 to 15 ft and the target may appear to be quite large. This necessitates a large, unobstructed area for data acquisition and a small target. Whitney reported that it was often time consuming to find the target and difficult to center the cross-hairs accurately. It was suggested that the passive target be replaced with a light source to improve target location and centering.

The use of the single theodolite has some distinct disadvantages. The major drawback is the time required to make readings. Whitney describes the use of the theodolite as "slow and fatiguing." As described above, use of the single theodolite also requires that additional length measurements be made. If not

made very accurately, these additional measurements can have a significant impact on the accuracy of the identified parameters.

3.3.2 Point Measurement

A number of measurement schemes have been developed that will determine three independent coordinates of a point on the end effector. These coordinates serve to locate the position of a target point in the world coordinate system. Active instrumentation systems using Cartesian or spherical coordinate systems as well as triangulation methods have been reported. Passive fixtures have also been used to locate the target point at predetermined locations in the workspace.

Perhaps the most straightforward approach to point measurement is the use of a coordinate measuring machine to determine the location of a target on the robot. The CMM is instrumented with a sensitive touch probe that causes the coordinates of the probe to be recorded when it touches an object. The target on the end effector is a uniform sphere. The touch probe is manipulated so that it touches several points on the surface of the target sphere. The center of the target sphere may then be determined from the coordinates of the points on the surface.

The use of a CMM for point measurement is appealing because the CMM is easy to use, requires only a simple target, and produces data that are easily interpreted. Unfortunately, most CMMs large enough to calibrate typical industrial robots are too expensive to justify for robot calibration alone.

In many instances, a CMM or precision measurement facility may be available but cannot accommodate a robot inside the facility. In this case, it may be possible to construct a fixture for the robot calibration and use the precision measurement facility to accurately determine the location of a number of points on the fixture. Veitschegger and Wu [5] have used such a procedure for calibration of a PUMA 560. Their method is based on the use of a fixture plate with a set of precisely positioned holes and an end effector with a pointing device. The fixture plate and the end effector are illustrated in Figure 3.16. A tool locating dowel, also illustrated in Figure 3.16, is placed in one of the holes. The pointer on the end effector is moved until the tip is centered a specific distance above the top of the tool locating dowel as determined by a feeler gauge. Since the location of the hole and the height of the tool locating dowel are precisely known, the coordinates of the tip of the end effector are also known.

If a precision measurement facility is available, a fixture technique such as the one described by Veitschegger and Wu represents a low cost means of obtaining the data required for calibration. Use of the fixture, however, can be quite time consuming if the robot does not have a "free" mode. Also, a precision fixture for calibrating a large robot may be too large itself to be practical. If the location of reference points on the fixture cannot be accurately determined or if the fixture is subject to significant levels of thermal or mechanical deformation, it will be of little use in the calibration process.

A unique point measurement system based on a spherical coordinate system

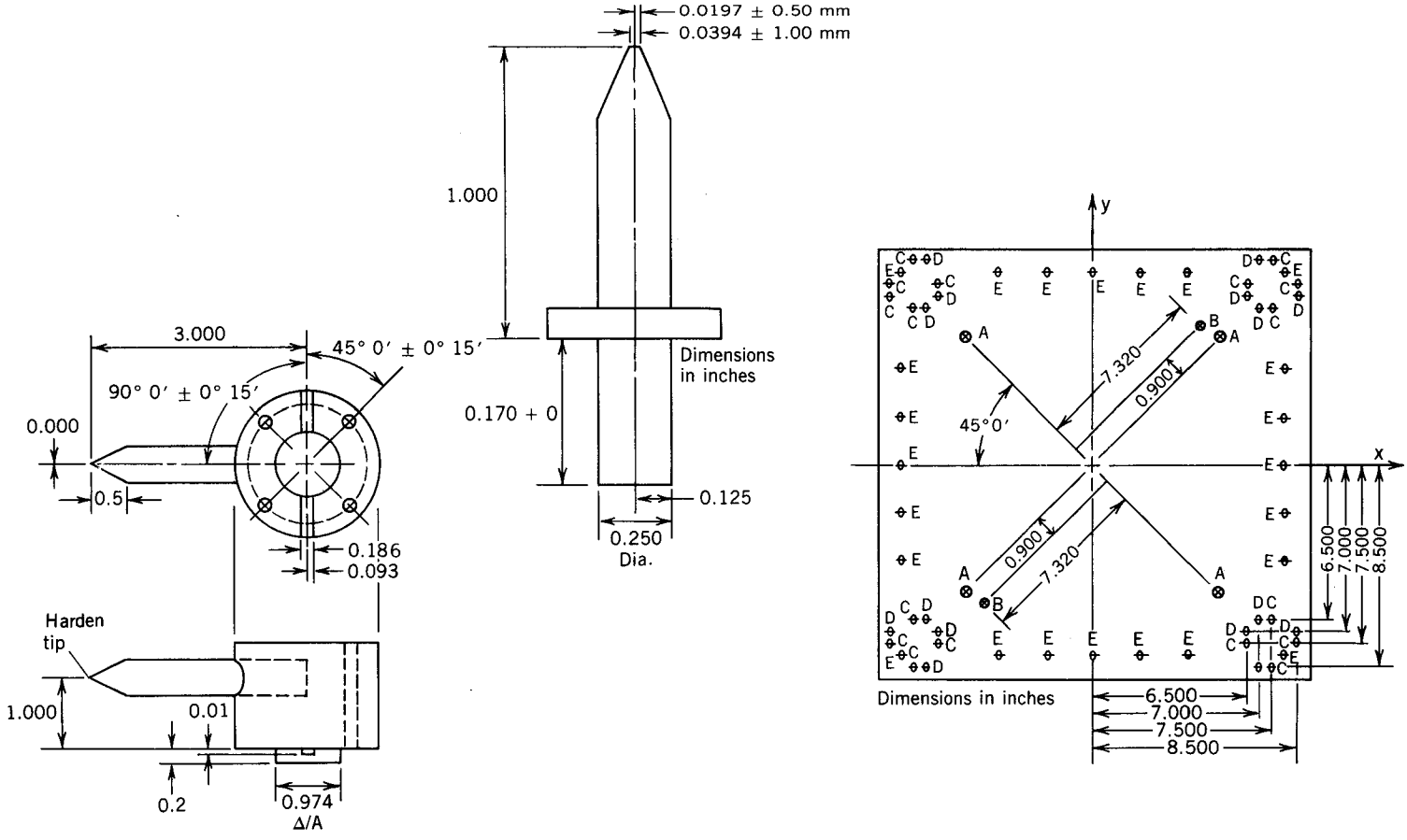


Figure 3.16. Fixture and end effector used by Veitschegger and Wu. Redrawn with permission of the authors [5]. Copyright © 1987 IEEE.

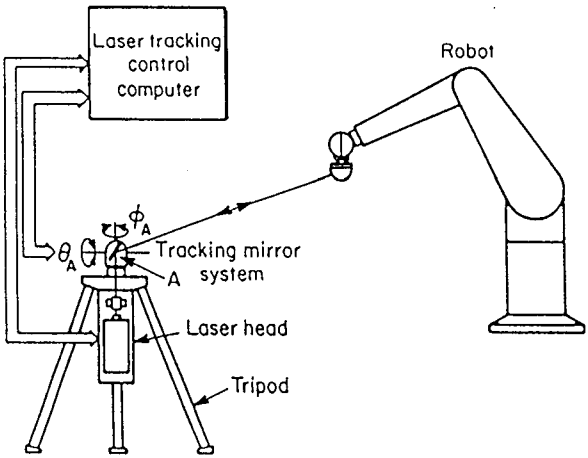


Figure 3.17. Point measurement system developed by Lau, Hocken, and Haight. Reprinted with permission of authors [6].

has been reported by Lau, Hocken, and Haight [6]. The system, which is illustrated in Figure 3.17, consists of a laser interferometer with a steerable beam. As illustrated in the figure, the beam leaves the laser and travels vertically until it strikes a mirror. The orientation of the mirror is precisely controlled by rotating the mirror about the horizontal and vertical axes. The mirror is oriented so that the beam will be directed toward a reflector mounted on the end effector of the manipulator. The returning beam is directed back toward the laser and then split several times. Two of the beam splitters are mounted orthogonally and direct portions of the returning beam onto photosensitive detectors. If the end effector begins to move, the alignment of the return beam will be modified. This is sensed by the photosensitive detectors and the mirror is rotated so that the return beam regains alignment. In this manner, the system will track the target throughout the workspace. In addition, the return beam is directed to an interferometer so that changes in distance from the mirror to the target may be determined very accurately. Since the mirror angles and the distance to the target are known precisely, the location of the target may be accurately determined. A version of this system is commercially available as the Smart 310 system from Leica.

This system offers several advantages. The primary advantage is that the system is totally automatic. Once the target has been located, the robot may be moved to any number of poses and the system will follow the motion. This would facilitate a totally automated calibration procedure. A second advantage is the precision. The use of the laser interferometer and high-resolution encoders on the mirror orientation axes gives excellent system accuracy. The disadvantages of this system are the same as those with any laser interferometer system. The interferometer simply counts interference patterns as the target moves away from some reference point. Some initialization procedure must be developed to establish the reference point. Also, if the beam is interrupted, the count will be destroyed and the measurement must be restarted.

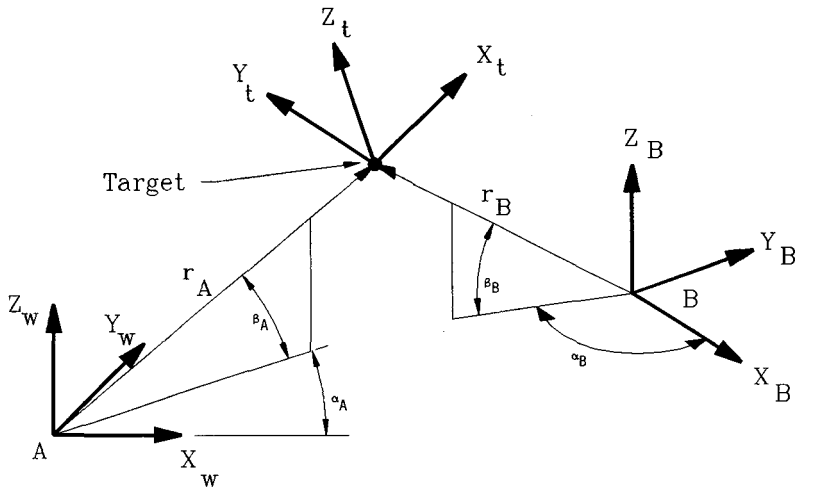


Figure 3.18. Triangulation.

A number of point measurement schemes make use of triangulation. Triangulation is based on the determination of the line of sight to a target from several points. The intersection of theses lines determines the location of the point. This is illustrated in Figure 3.18. The observer at point A sights the target and records the horizontal and vertical angles at his location (α_A and β_A). The observer at point B records α_B and β_B in a similar manner. The world coordinate system, XYZ_w, is established at point A and a unit vector, η_A, is defined as being the unit vector along the line of sight. A separate coordinate system, XYZ_B, is defined at point B and a unit vector along the line of sight at B is defined as η_B. Using the definition of α and β in the figure, the unit vectors will be given by

$$\eta_A = \begin{bmatrix} \cos(\beta_A)\cos(\alpha_A) \\ \cos(\beta_A)\sin(\alpha_A) \\ \sin(\beta_A) \\ 1 \end{bmatrix} \tag{3.6}$$

and

$$\eta_B = \begin{bmatrix} \cos(\beta_B)\cos(\alpha_B) \\ \cos(\beta_B)\sin(\alpha_B) \\ \sin(\beta_B) \\ 1 \end{bmatrix} \tag{3.7}$$

where η_A is measured in the XYZ_w coordinate system and η_A is measured in the XYZ_B coordinate system. Since both unit vectors are directed at the same target,

two equivalent expressions for the location of the target may be written as

$$r_A \eta_A = \mathbf{T}_{AB} \eta_B r_B \tag{3.8}$$

where r_A and r_B are the respective distances from the origin of each coordinate system to the target point and $\mathbf{T}_{AB}$ is the transformation from the XYZ_B system to the world system. The problem now is to determine the values of r_A and r_B given the measured angles. If Equation 3.8 is expanded and terms rearranged, the following equation results.

$$\begin{bmatrix} c\beta_A c\alpha_A & -T_{11}c\beta_B c\alpha_B - T_{12}c\beta_B s\alpha_B - T_{13}s\beta_B \\ c\beta_A s\alpha_A & -T_{21}c\beta_B c\alpha_B - T_{22}c\beta_B s\alpha_B - T_{23}s\beta_B \\ s\beta_A & -T_{31}c\beta_B c\alpha_B - T_{32}c\beta_B s\alpha_B - T_{33}s\beta_B \end{bmatrix} \mathbf{r} = \begin{bmatrix} T_{14} \\ T_{24} \\ T_{34} \end{bmatrix} \tag{3.9}$$

where $s\alpha_A$ represents $\sin(\alpha_A)$, $c\alpha_A$ represents $\cos(\alpha_A)$, and so forth. This equation may be expressed symbolically as follows.

$$\mathbf{Cr} = \mathbf{v} \tag{3.10}$$

Only three independent values need to be measured to determine the location of the point. Since we have measured four angles, it is reasonable to expect an overdetermined set of equations such as in Equation 3.10. We may simply choose two of the three equations to solve for r_A and r_B. If the horizontal and vertical angles were measured perfectly, this would be an acceptable approach. Since any real measurement system will contain some error, it is desirable to use all three equations and determine a "best fit" value for **r**. Using least squares (see Section 4.3), **r** will be given by

$$\mathbf{r} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{v} \tag{3.11}$$

To continue to minimize the effect of measurement errors, the target may be sighted from additional points. In this case, Equation 3.10 may be expanded to include the additional data. Likewise, Equation 3.11 may be modified to include the additional measurements.

Having determined **r**, the pose of the end effector is now given by

$$\begin{bmatrix} ? & ? & ? & r_A c\beta_A c\alpha_A \\ ? & ? & ? & r_A c\beta_A s\alpha_A \\ ? & ? & ? & r_A s\beta_A \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

where the question marks represent elements of the pose that are not determined by the measurement system. It is important to note the factors effecting the accuracy of the solution for **r**. Certainly if the horizontal or vertical angles are in error, the result will be incorrect. Of equal importance, however, is the

relationship between the XYZ_B and the world system. If the location or orientation of the XYZ_B system is not known precisely, the resulting solution for $\mathbf{r}$ will be in error. This is important since many of the devices used to determine the horizontal and vertical angles are not capable of measuring distance or the orientation of one device with respect to another.

Judd and Knasinski [7] conducted a calibration of an Automatix AID-900 robot using triangulation. Two theodolites were used to determine the location of targets on a specially designed end effector. The process for determining the relative position and orientation of the theodolites was not discussed in this paper. Chen and Chao [8] also reported the calibration of a PUMA 760 robot using triangulation. In this work, three theodolites were used to locate a target point on the robot end effector. As in the paper by Judd and Knasinski, there was no mention of how the relative position and orientation of the theodolites were determined.

Jarvis [9] has reported a three-step procedure for calibration of theodolites to be used for triangulation. The first step is to sight a target with each theodolite at a number of locations along a straight line. The target should be moved precisely along a line and the distance between each point must be known. The orientation of each theodolite with respect to this line may then be determined. The second step is to sight a number of arbitrarily located points with both theodolites simultaneously. Finally, the data taken in the line and point sightings may be used to compute the matrix $\mathbf{T}_{AB}$, which Jarvis describes as the baseline.

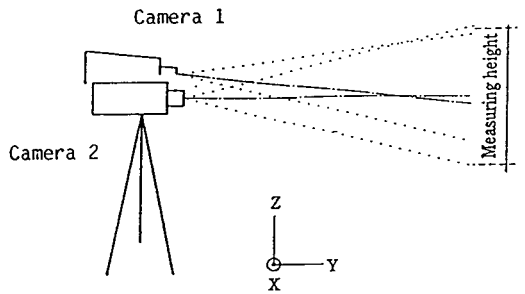
Commercially available systems that are based on triangulation are available. A typical one is offered by the Selspot company. The system is based on the use of camera-type devices that are described in Section 3.1. For point measurement, cameras are set up at two different locations so that each views a significant portion of the workspace. This is illustrated in Figure 3.19. Each camera returns readings that are equivalent to the horizontal and vertical angles to the target. The system also has a calibration fixture and program so that the position and orientation of each camera with respect to a world coordinate system may be determined.

Another approach to point measurement, which is similar to triangulation, is based on the measurement of distance to a target from several points located in the workspace. This approach is illustrated in Figure 3.20. The procedure begins by establishing at least three measurement points whose locations in the world coordinate system, XYZ_w, are known precisely. The distance from each of these points to a target on the robot end effector is then measured. If these distances are given by r_a, r_b, and r_c and the locations of the measurement points are given by $\mathbf{r}_{pa}$, $\mathbf{r}_{pb}$, and $\mathbf{r}_{pc}$, the location of the target point will be given by any of the following three equations

$$\mathbf{r}_{pa} + r_a \boldsymbol{\eta}_a = \mathbf{r}_t \tag{3.13}$$

$$\mathbf{r}_{pb} + r_b \boldsymbol{\eta}_b = \mathbf{r}_t \tag{3.14}$$

$$\mathbf{r}_{pc} + r_c \boldsymbol{\eta}_c = \mathbf{r}_t \tag{3.15}$$



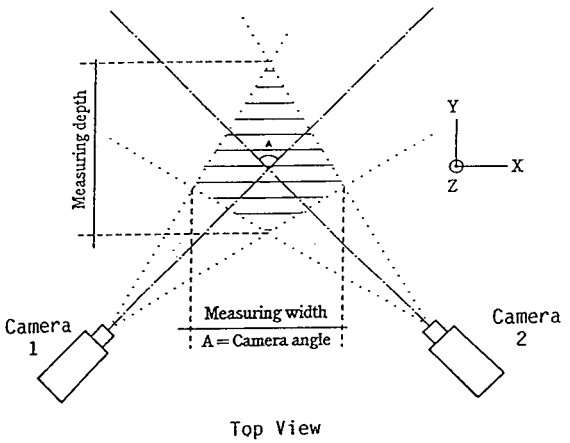


Figure 3.19. Selspot system. Reprinted with permission of Selcom, Inc., Southfield, MI.

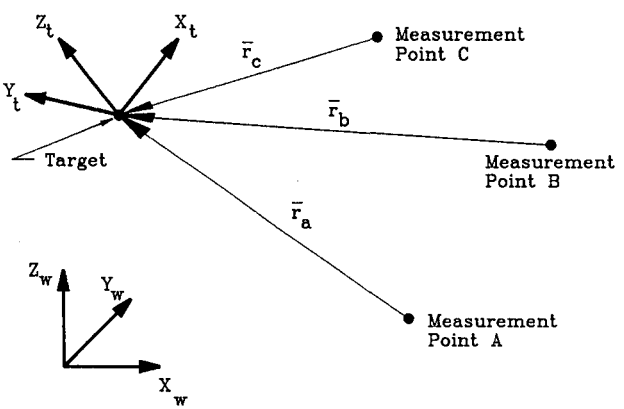


Figure 3.20. Point location by distance measurement.

where $\boldsymbol{\eta}_i$ is a unit vector directed from the ith measurement point toward the target and $\mathbf{r}_t$ is a vector from the origin of the world coordinate system to the target point. In these equations, the unknown values are the directions to the target, $\boldsymbol{\eta}_i$, and the location of the target, $\mathbf{r}_t$. The vector to the target, $\mathbf{r}_t$, may be eliminated from Equations 3.14, 3.15, and 3.15 as follows:

$$r_a\boldsymbol{\eta}_a - r_b\boldsymbol{\eta}_b = \mathbf{r}_{pb} - \mathbf{r}_{pa} \tag{3.16}$$

$$r_c\boldsymbol{\eta}_c - r_b\boldsymbol{\eta}_b = \mathbf{r}_{pb} - \mathbf{r}_{pc} \tag{3.17}$$

Since the unit vectors, $\boldsymbol{\eta}_a$, $\boldsymbol{\eta}_b$, $\boldsymbol{\eta}_c$, are unknown, these vector equations represent six scalar equations in nine unknowns. Three additional equations result from the knowledge that the $\boldsymbol{\eta}$s are unit vectors.

$$\eta_{ax}^2 + \eta_{ay}^2 + \eta_{az}^2 = 1 \tag{3.18}$$

$$\eta_{bx}^2 + \eta_{by}^2 + \eta_{bz}^2 = 1 \tag{3.19}$$

$$\eta_{cx}^2 + \eta_{cy}^2 + \eta_{cz}^2 = 1 \tag{3.20}$$

These nine scalar equations are nonlinear and are typically solved numerically to yield values for $\boldsymbol{\eta}_a$, $\boldsymbol{\eta}_b$, and $\boldsymbol{\eta}_c$. The location of the target point, $\mathbf{r}_t$, is then given by any of the Equations 3.13, 3.14, or 3.15. The end effector pose may then be written as

$$\begin{bmatrix} ? & ? & ? & r_a\eta_{ax} \\ ? & ? & ? & r_a\eta_{ay} \\ ? & ? & ? & r_a\eta_{az} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

where the question mark represents elements of the pose that are not determined by the measurement system.

As with the triangulation method, small errors in the measurement can have a significant effect on the estimated location of the target point. To minimize this effect, measurements may be made from more than three points in the workspace. If this is done, an overdetermined set of nonlinear equations results that may be solved so as to minimize the square of the total error. Such extra measurements tend to offset the effect of measurement noise on the final solution. Also, the position of the various measurement points in the world coordinate system must be accurately known. This is different than the triangulation technique where both position and orientation were required. The lack of a requirement for relative orientation of the measurement devices significantly simplifies the calibration of the measurement system.

A measurement system based on the distance approach is commercially available from Chesapeake Laser Systems. This system, illustrated in Figure 3.21, consists of a low power, helium–neon laser whose beam is split and directed to some number of tracking modules. Each beam is instrumented with an interferometer so that the distance to the retroreflector on the target is accurately known.

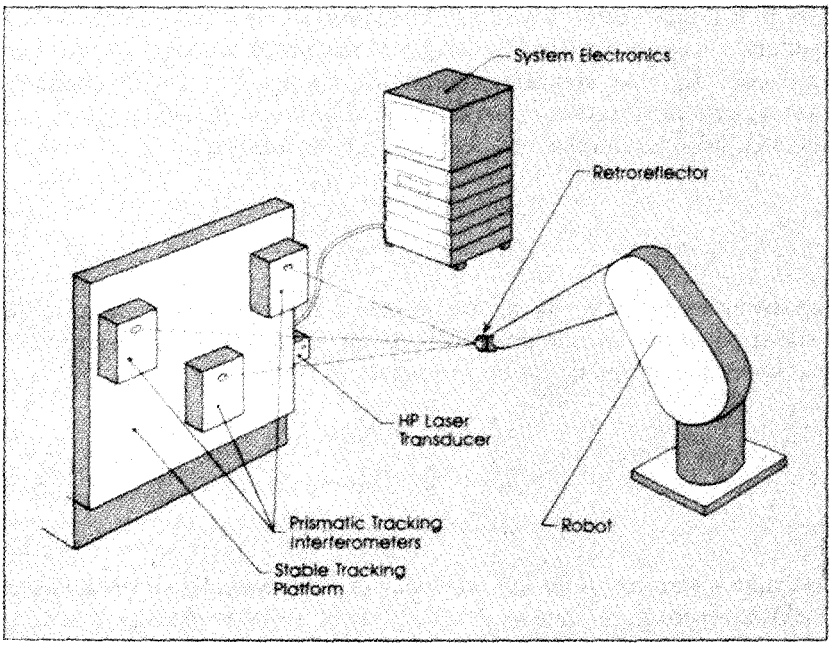


Figure 3.21. The CMS-1000 point location system. Reprinted with permission from product literature for Chesapeake Laser Systems, Inc., laser coordinate measuring system, CMS-1000, Copyright 1989.

The tracking modules are designed so that the beams are always pointed at the retroreflector. The manufacturer states that the position of the target may be determined to an accuracy of 1 μm (0.00004 in.) over a 10-m workspace.

Another measurement system based on the same approach was reported by Stone, Sanderson, and Neumann [10, 11]. In this work, the target consisted of an acoustic emitter with microphones located at a number of measurement points throughout the workspace. Time of flight of the acoustic signal was used to determine the distance from the target to each of the microphones. The authors reported a system resolution of 0.008 in. Although this is significantly less resolution than is provided by the Chesapeake system, it should be noted that the sonic system is much less expensive.

3.3.3 Partial Pose Measurement

Several measurement systems have been developed that will return some information about the orientation of the end effector as well as the position of a target point. We have defined these as "partial pose" systems because only one or two components of the end effector orientation is determined.

The system developed by Lau, Hocken, and Haight [6] for point measurement was extended to yield orientation information. As illustrated in Figure 3.22, the retroreflector on the end effector was replaced with a mirror whose orientation

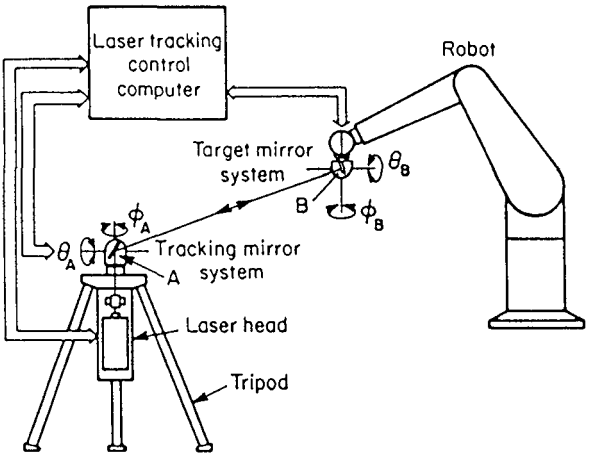


Figure 3.22. Partial pose system developed by Lau, Hocken, and Haight. Reprinted with permission of authors [6].

is under servo control. By orienting the mirror so that the beam is always directed back at the laser, two components of end effector orientation may be determined.

Another partial pose method has been developed by Tang [12]. This approach was developed in an attempt to simplify the measurement process by creating a simple fixture arrangement. The fixture proposed by Tang consists of a flat plate with a grid of accurately located points on the plate. This may be purchased commercially as an optical table. These tables are extremely flat and have a number of tapped holes that are accurately located in a grid on the table. An end effector is designed with a flat surface at some angle to the last axis of the robot. The end effector also is equipped with a reference point that may be located at the known locations on the table. When the flat surface on the end effector is against the table, one component of orientation is known. Also when the reference point on the end effector is located over a table point, the position of the end effector is known. These partial poses may then be used to complete the calibration process.

A point measurement device may be used repeatedly on a fixture to determine some additional components of pose. This technique was employed by Borm and Menq [13] in their calibration of an RM501 robot. The end effector used in this work was a simpler cylinder and the measurement device was a Sheffield CMM. The CMM was used to locate four points on the cyclinder face and four points around the circumference of the cylinder. These data allowed five components of the end effector pose to be determined with high precision.

3.3.4 Complete Pose Measurement

We know of no active device that will directly measure the complete pose of an object in space. The complete pose of an end effector may be determined by

locating at least three points on the end effector whose relative position is known. For example, assume that an end effector is equipped with three target points and that the location of these points in the tool coordinate system is given by $\mathbf{r}_{1t}$, $\mathbf{r}_{2t}$, and $\mathbf{r}_{3t}$. If the point measurement system has determined the location of each point in the world system, $\mathbf{r}_{1w}$, $\mathbf{r}_{2w}$, and $\mathbf{r}_{3w}$, the end effector pose may be determined. If the unknown transformation from the tool coordinate system to the world coordinate system is defined as $\mathbf{T}_{wt}$, the following equation may be written:

$$[\mathbf{r}_{1w}, \mathbf{r}_{2w}, \mathbf{r}_{3w}] = \mathbf{T}_{wt}[\mathbf{r}_{1t}, \mathbf{r}_{2t}, \mathbf{r}_{3t}] \tag{3.22}$$

A solution for $\mathbf{T}_{tw}$ may be obtained by defining two additional points as follows:

$$\mathbf{r}_{4w} = (\mathbf{r}_{2w} - \mathbf{r}_{1w}) \times (\mathbf{r}_{3w} - \mathbf{r}_{1w}) + \mathbf{r}_{1w} \tag{3.23}$$

$$\mathbf{r}_{4t} = (\mathbf{r}_{2t} - \mathbf{r}_{1t}) \times (\mathbf{r}_{3t} - \mathbf{r}_{1t}) + \mathbf{r}_{1t} \tag{3.24}$$

$$\tag{3.25}$$

These points may be added to the equation above to yield the following:

$$[\mathbf{r}_{1w}, \mathbf{r}_{2w}, \mathbf{r}_{3w}, \mathbf{r}_{4w}] = \mathbf{T}_{wt}[\mathbf{r}_{1t}, \mathbf{r}_{2t}, \mathbf{r}_{3t}, \mathbf{r}_{4t}] \tag{3.26}$$

Since all matrices in the above equation have the same dimension, we may post multiply to achieve the following result:

$$\mathbf{T}_{tw} = [\mathbf{r}_{1w}, \mathbf{r}_{2w}, \mathbf{r}_{3w}, \mathbf{r}_{4w}][\mathbf{r}_{1t}, \mathbf{r}_{2t}, \mathbf{r}_{3t}, \mathbf{r}_{4t}]^{-1} \tag{3.27}$$

In the work by Judd and Knasinski [7], a procedure similar to the one above was used to determine a complete pose. An end effector was designed with three targets and the location of each target in the world coordinate system was determined at each pose. This information was then used as described above to obtain the complete pose information. A similar approach was used by Mooring and Padavala [14] in their calibration of a PUMA 560 robot. A detailed example of a calibration using this measurement approach is given in the Case Study in Chapter 6 of this book.

Several fixture systems have been devised that will define complete poses for a robot. A passive fixturing system has been reported by Hayati and Roston [15]. This system consists of a number of keyed fixtures that are accurately located in the workspace of a PUMA 250 robot. The end effector is designed so that when mated with the fixture, the complete pose is known. This procedure works quite well because the PUMA 250 is a physically small robot and it has a "free" mode that allows manual insertion of the end effector into the fixtures.

In those instances where manual insertion into a fixture is not possible or the calibration process is to be automated, a set of active fixtures may be used. An active fixture is one that makes use of short-range measurement devices to

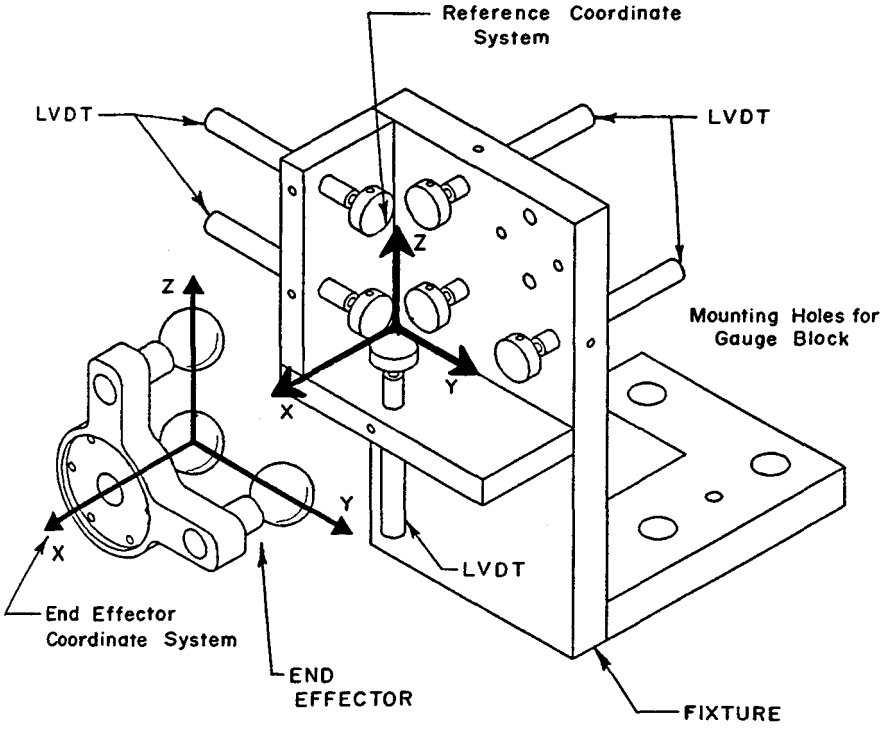


Figure 3.23. Active pose fixture.

determine the location of an end effector in a local coordinate system. The relationship of each of the fixtures in the world coordinate system must be accurately determined with another measurement device. Active fixtures are also often used for robot repeatability studies and various designs have been reported by [16, 17]. A fixture typical of these is illustrated in Figure 3.23. As shown in the figure, the robot end effector has a set of three spheres that are inserted in an array of short-range displacement transducers. In this case these transducers are LVDTs but capacitance probes and dial indicators have also been reported.

A unique and quite clever passive measurement scheme has been reported by Bennett and Hollerbach [18, 19]. In their technique, enough extra joints are added to the robot structure to make the mechanism a single loop, closed kinematic chain with more than 1 degree of freedom. This may be accomplished by adding a link and an additional joint between the end effector and ground on a single robot or by rigidly connecting the end effectors of two open chain robots. The additional degrees of freedom mean that the closed loop is free to take on an infinite number of configurations as long as the loop closure and joint travel constraints are not violated. Bennett and Hollerbach demonstrated that the use of the readings from the joint transducers in such a case is sufficient to identify a large percentage of the kinematic structure of the closed chain.

3.4 CONCLUSION

As indicated by the large number of measurement devices and approaches to data collection that have been described above, there is no "best" measurement system for robot calibration. In most cases, the selection of a measurement scheme is a tradeoff between the accuracy desired and the cost of the system. In some cases, the cost can be mitigated by borrowing or renting a high accuracy active system such as a laser interferometer or a CMM and using it to build or verify a fixturing system that consists of less expensive components. In any event, the most desirable system will be the one that represents the best compromise between cost, ease of use, and precision for a given calibration task.

REFERENCES

[1] Ernest O. Doebelin. *Measurement Systems: Application and Design.* McGraw-Hill, New York, 1983.

[2] Richard D. Klafter, Thomas A. Chmielewski, and Michale Negin. *Robotic Engineering, An Integrated Approach.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[3] M. A. R. Cooper. *Modern Theodolites and Levels,* 2nd ed. Granada, London, 1982.

[4] D. E. Whitney, C. A. Lozinski, and J. M. Rourke. Industrial robot forward calibration method and results. *Journal of Dynamic Systems, Measurement, and Control,* 108(1):1–8, March 1986.

[5] W. K. Veitschegger and Chi-Haur Wu. A method for calibrating and compensating robot kinematic errors. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation,* pp. 39–44, April 1987.

[6] K. Lau, R. J. Hocken, and W. C. Haight. Automatic laser tracking interferometer system for robot metrology. *Precision Engineering,* 8(1), January 1986.

[7] Robert P. Judd and A. B. Knasinski, A technique to calibrate industrial robots with experimental verification. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation,* pp. 351–357, April 1987.

[8] J. Chen and L. M. Chao. Positioning error analysis for robot manipulators with all rotary joints. In *Proceedings of 1986 IEEE International Conference on Robotics and Automation,* pp. 1011–1016, April 1986.

[9] John F. Jarvis. Microsurveying: Towards robot accuracy. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation,* pp, 1660–1665, April 1987.

[10] Henry W. Stone and Arthur C. Sanderson. A prototype arm signature identification system. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation,* pp. 175–182, April 1987.

[11] Henry W. Stone, Arthur C. Sanderson, and Charles P. Neuman. Arm signature identification. In *Proceedings of 1986 IEEE International Conference on Robotics and Automation,* pp. 41–48, April 1986.

[12] Stanley C. Tang and Ching-Cheng Wang. Correction for robot positioning errors an approach to improve robot applicability and productivity. In *Proceedings of the IXth ICPR,* pp. 1947–1953, August 1987.

[13] Jin-Hwan Borm and Chia-Hsiang Menq. Experimental study of observability of parameter errors in robot calibration. In *Proceedings of 1989 IEEE Conference on Robotics and Automation*, pp. 587–592, IEEE, May 1989.

[14] B. W. Mooring and S. S. Padavala. The effect of model complexity on robot accuracy. In *Proceedings of 1989 IEEE Conference on Robotics and Automation*, pp. 593–598, IEEE, May 1989.

[15] Samad A. Hayati and Gerald P. Roston. In *Recent Trends in Robotics: Modeling, Control, and Education*, Inverse kinematic solution for near simple robots and its application to robot calibration, pp. 41–50. Elsevier Science Publishing Co., Amsterdam, 1986.

[16] B. W. Mooring and T. J. Pack. Aspects of robot repeatability. *Robotica*, 5:223–230, 1987.

[17] Paul G. Ranky and Michael Wodzinski. Robot pose error testing. In *Tutorial Presented at 1987 IEEE International Robotics and Automation Conference*, 1987.

[18] David J. Bennett and John M. Hollerbach. Self calibration of single-loop, closed kinematic chains formed by dual or redundant manipulators. In *Proceedings of the 27th Conference on Decision and Control*, pp. 627–629, Austin, TX, December 1988.

[19] David J. Bennett and John M. Hollerbach. Identifying the kinematics of robots and their tasks. In *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, pp. 580–586, Scottsdale, AZ, May 1989.

CHAPTER 4

PARAMETER IDENTIFICATION FOR ROBOT CALIBRATION

Once a valid manipulator model has been developed and a set of measurement data has been collected, the next task is to determine the set of model parameters that causes the poses computed from the model to most closely match the measured data. This process is referred to as the *identification* step. The topic of parameter identification has been studied in depth for a number of years as it applies to such fields as control theory and dynamic systems modeling. Many of the standard parameter identification techniques are directly applicable to the manipulator calibration problem. In this chapter, we will review the applicable parameter identification methods and demonstrate their use specifically for the calibration problem.

4.1 IDENTIFICATION ISSUES

Any system identification problem requires three basic ingredients:

1. A mathematical model.
2. A set of variables that needs to be estimated.
3. Measured data.

Typically, the variables to be estimated are unknown parameters in the given mathematical model. It is also usually assumed that the data have some level of measurement noise. The primary objective of the identification procedure is to extract information from the observed data while rejecting the noise. Most identification techniques attempt to determine the variable set that will optimize

some performance index. The difference in the various procedures comes in the type of model that is used or in the assumptions that are made about the noise. Identification techniques are classified in the following manner:

1. Deterministic vs Stochastic—depending on whether or not probabilistic models for process and measurement noise are utilized.
2. Recursive vs Nonrecursive—depending on whether or not the whole set of observed data is saved and processed in its entirety, or used sequentially, thus generating a sequence of estimates that is based on a growing set of data. In a recursive estimation, at every step the estimate is based on the optimal estimate at the previous step (representing all the past data) and the new set of measurements only.
3. Linear vs Nonlinear—depending on the type of mathematical model that is used.

As described in Chapter 1, the process of manipulator calibration consists of the modeling, measurement, identification, and correction steps. Modeling refers to the choice of a functional relationship between the robot parameters and the resulting pose of the end effector as described in Chapter 2. The model selected should account for all the factors considered to be significant in contributing to robot accuracy. Since this chapter is concerned with identification rather than modeling, we make use of the standard Denavit–Hartenberg (DH) notation [23] in most of the following derivations and simulation examples. Many of the conclusions drawn using this notation apply equally well to other kinematic models.

Physical data are then collected from measurements done on the robot that needs to be calibrated. This process is described in Chapter 3. These data contain information relating the input of the model (the readings of the joint transducers) to the output of the model (the robot pose). The mathematical process of using the data collected to identify the coefficients of the model is the third step in calibration. An important consideration during identification is the expected error in the identified coefficients because of noise in the measurement process.

Examination of the literature on robot calibration shows that a variety of numerical methods have been used to identify geometric and nongeometric parameters. The use of these techniques is sometimes not as straightforward as one might think, and some care is required to frame the problem in such a way as to make the identification fast and accurate. In this chapter we examine, through theoretical derivation and simulation examples, those factors that need to be considered in identifying the kinematic parameters of a robot manipulator. Such factors include the type of identification algorithm used, the initial estimate of the required parameters, the effect of measurement accuracy and noise, encoder resolution and noise, the number of poses measured, the selection of the measurement configurations, and the range of motion of the robot joints during the observations. Understanding of all these factors enhances the design and planning of robot calibration systems.

It will be helpful at this point to consider two similar robots, A and B. The identification problem is to estimate the model of robot B given the model of robot A and a set of measurements made on robot B. Physically robots A and B may be the same machine where robot A model is the perfect or *nominal* model, and robot B model is the *actual* model. On the other hand, robots A and B may sometimes be two separate machines that are nominally the same but practically dissimilar due to machining and assembly tolerances. In the latter case, it is assumed that robot A is the robot for which the application program has been programmed. Robot B replaces robot A in running the desired application.

There are two philosophies in relating the actual model, robot B, to the nominal, robot A. One is parametric in nature and the other is geometric. As is well known, the construction of the robot kinematic model, regardless of modeling convention, always starts by specifying the robot joint axis lines in an arbitrary robot configuration. Link parameters such as common normals and twist angles are then determined from analysis of the relative location of pairs of adjacent joint axes. This is the idea behind the geometric approach.

The parametric identification approach, on the other hand, is to assume additive errors in the robot link parameters. The identification problem is then that of finding the vector of the kinematic parameter errors. This class of techniques breaks into two subclasses, based on whether or not one chooses to linearize the actual manipulator kinematic transformations. By expanding the product of homogeneous transformations of the actual robot, ignoring second-order products of error parameters, one obtains the identification Jacobian, or the linearized error model as was shown in Chapter 2. Such error models are the basis for various *linear least-squares* techniques discussed more fully in Section 4.3.

The second approach to find the kinematic parameter errors is by fitting a nonlinear regression model. Such *nonlinear least-squares* techniques are the subject of Section 4.4.

The geometric identification philosophy, mentioned above, starts with the identification of the joint axes themselves; namely, identification of the 3D line equations that constitute the set of all robot joint axes at a given robot configuration. From the identified axes, the actual kinematic model may be constructed according to whatever modeling convention. For instance, one may extract the actual DH link parameters directly from the identified joint axes. Section 4.6 covers in detail the techniques for estimating the robot joint axes. This section also includes a description of extraction of the kinematic parameters from the identified joint axes.

There are several practical questions that are crucial for the identification phase in robot calibration:

- What is the relationship between the parameter estimation error and the accuracy of the calibration sensor? Given a required robot accuracy, can minimum requirements for the calibration equipment be specified?

- Can one infinitely improve on the calibration accuracy by taking more and more measurements? Obviously not, so the question is restated as follows: What is the relationship between the lower bound on the calibration error and the robot repeatability measures?
- How many measurements need to be taken to achieve a prespecified identification accuracy? This question is of crucial importance in those cases in which each calibration measurement is done by creating an interaction between a robot tool and a fixed calibration fixture located in the robot workspace. To avoid excessive fixturing costs, it is important to determine the smallest number of measurements that will result in an accurate calibration.
- Having different identification algorithms that vary in their modeling requirements, does additional modeling beyond kinematic modeling, such as probabilistic characterization of measurement noise and unknown robot parameters, really promise a significant improvement of the calibration accuracy?

While linear least-squares and nonlinear least-squares algorithms are commonly used as simple practical ways of estimating the unknown parameters from the measured data, one needs to resort to Estimation Theory and to analysis tools such as Kalman filters for some useful formulas that provide relationships between estimation error and parameters such as measurement noise covariance or the number of measurements taken. Such analysis tools can then provide better insight as to what is the best way of working with the "practical" algorithmic methods.

It is important to stress that the parametric and geometric identification approaches are not completely disjoint. For instance, the identification of the robot joint axes line equations alone will not suffice to determine the joint variable offsets. Furthermore, the identification of joint axes can be done only for axes about which an actual motion (either rotary or linear) can be performed during the measurement phase. As such the robot base and tool transformations parameters need to be found parametrically. The last section in this chapter, Section 4.7, is devoted to base and tool parameter identification issues.

4.2 MATHEMATICAL FORMULATION FOR IDENTIFICATION OF ROBOT KINEMATICS

The forward kinematic model of a robot manipulator describes the position and orientation of the tool frame attached to the end of the manipulator in terms of the base frame. Let the base link fixed to the ground be numbered 0. The tool frame at the most distal link is numbered n. Each link i has a coordinate frame $O_{x_i y_i z_i}$ attached to it. The matrix $\mathbf{A}_i^{i-1}$ is a homogeneous transformation representing the position and orientation of the frame i relative to frame $i-1$.

Thus, using the DH notation:

$$\mathbf{A}_i^{i-1} = \mathbf{A}_i^{i-1}(\mathbf{p}_i) \qquad \mathbf{p}_i = [\theta_i\ r_i\ l_i\ \alpha_i]^T \tag{4.1}$$

where $\mathbf{p}_i$ is the parameter vector for joint i.

The kinematic equation of the robot manipulator is obtained by the consecutive homogeneous transformations from the last frame back to the base frame. Thus

$$\mathbf{T}_n^0 = \mathbf{T}_n^0(\mathbf{k}) = \mathbf{A}_1^0\mathbf{A}_2^1 \cdots \mathbf{A}_n^{n-1} = \prod_{i=1}^{n} (\mathbf{A}_i^{i-1}) \tag{4.2}$$

where $\mathbf{k} = [\mathbf{p}_1\ \mathbf{p}_2 \cdots \mathbf{p}_n]^T$ is the parameter vector for the manipulator.

Let $\delta\mathbf{p}_i = [\delta\theta_i\ \delta r_i\ \delta l_i\ \delta\alpha_i]^T$ where $\delta\mathbf{p}_i$ is the link parameter error vector. If joint i is revolute, then $\delta\theta_i$ is the encoder offset at that joint. If joint i is prismatic, then δr_i is the encoder offset at that joint. The exact link transformation $\mathbf{B}_i^{i-1}$ is

$$\mathbf{B}_i^{i-1} = \mathbf{A}_i^{i-1} + d\mathbf{A}_i \qquad d\mathbf{A}_i = d\mathbf{A}_i(\delta\mathbf{p}_i) \tag{4.3}$$

The exact manipulator transformation $\mathbf{T}_{B,n}^0$ is

$$\mathbf{T}_{B,n}^0 = \prod_{i=1}^{n} (\mathbf{B}_i^{i-1}) \tag{4.4}$$

The additive error transformation, $d\mathbf{T}$, is defined as

$$d\mathbf{T} = \mathbf{T}_{B,n}^0 - \mathbf{T}_n^0 \tag{4.5}$$

Let the vector $\mathbf{q}$ be the vector of joint variables (θ_i or r_i) and $\delta\mathbf{k} = [\delta\mathbf{p}_1\ \delta\mathbf{p}_2 \cdots \delta\mathbf{p}_n]^T$ be the manipulator parameter error vector.

Let $d\mathbf{T} = \mathbf{T}_n^0\Delta\mathbf{T}$. Here $\Delta\mathbf{T}$ is the transformation representing change in manipulator transformation $\mathbf{T}_n^0$ expressed in coordinate frame n. The manipulator transformation error $\Delta\mathbf{T}$ is a nonlinear function of the manipulator parameter error $\delta\mathbf{k}$. Thus $\Delta\mathbf{T} = \Delta\mathbf{T}(\mathbf{q}, \delta\mathbf{k})$. For a more rigorous treatment of differential transformations and their relationship to the calibration problem, see Section 4.7.

The robot identification problem addresses the issue of determining $\delta\mathbf{k}$ from measurements containing information about $\Delta\mathbf{T}$. Expanding Equation 4.4 and ignoring second-order products, we can obtain the identification Jacobian relating the change in manipulator transformation to the parameter error vector as

$$\delta\mathbf{T}_j = \mathbf{J}_j\delta\mathbf{k} \qquad j = 1, 2, \ldots, m \tag{4.6}$$

where $\delta\mathbf{T}$ is the differential translation and rotation vector (discussed in Chapter 2) and m is the number of observations. The number of observations is chosen to be large enough so as to overcome the effects of noise and uncertainty.

Equation 4.6 represents an overdetermined linear system that may be "solved" using the least-squares approach of the pseudoinverse (normal equations) as

$$\delta \mathbf{k} = [(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T]\delta\mathbf{T} \tag{4.7}$$

where $\mathbf{J} = [\mathbf{J}_1\ \mathbf{J}_2 \cdots \mathbf{J}_m]^T$ and $\delta\mathbf{T} = [\delta\mathbf{T}_1\ \delta\mathbf{T}_2 \cdots \delta\mathbf{T}_m]^T$. Equation 4.7 may be applied iteratively and will converge if the kinematic errors are small enough. This approach to identification will be referred to as the *linear least-squares method* and is described in Section 4.3.

The second approach is to view the problem as one of fitting a nonlinear regression model. The form of the nonlinear model for the exact manipulator transformation from Equation 4.4 is

$$\mathbf{x}_j = \mathbf{f}(\mathbf{q}_j, \mathbf{k}) \qquad j = 1, 2, \ldots, m \tag{4.8}$$

where $\mathbf{x}$ is the exact manipulator end-effector pose corresponding to $\mathbf{B}_n^0$, and m is the number of observations. The residuals for such a model are

$$\mathbf{e}_j(\delta\mathbf{k}) = \mathbf{z}_j - \mathbf{f}(\mathbf{q}_j, \mathbf{k}) \qquad j = 1, 2, \ldots, m \tag{4.9}$$

where $\mathbf{z}_j$ is the vector representing the measured pose of the robot at position j. The least squares estimate of $\mathbf{k}$ is the value of $\mathbf{k}$ that minimizes the function

$$L = \sum_{j=1}^{m} \mathbf{e}_j{}^T\mathbf{e}_j \tag{4.10}$$

Such an approach to identification will be referred to as the *nonlinear least-squares* method and is discussed in Section 4.4

The vector formulation in Equation 4.8 of the robot kinematic model is helpful in mathematically formulating some of the research issues that correspond to the calibration identification phase.

The first such issue is the relationship between the vector of joint variables $\mathbf{q}_j$ and the vector of joint position transducer readings $\boldsymbol{\eta}_j$ at measurement position j. Rewriting Equation 4.8 in terms of the joint readings $\boldsymbol{\eta}_j$ can be done as follows:

$$\mathbf{x}_j = \mathbf{f}(\boldsymbol{\eta}_j, \boldsymbol{\mu}, \mathbf{a}) \tag{4.11}$$

where $\boldsymbol{\eta}_j$ is the n-vector of joint tranducer readings in the case of an n degree of freedom manipulator, $\boldsymbol{\mu}$ is the vector of coefficients in the relationships between the joint transducers and the actual joint displacements, and $\mathbf{a}$ is the vector of coefficients in the kinematic model that is being used. Specializing Equation 4.11 to each of the robots A and B mentioned earlier provides:

$$\mathbf{x}_A = \mathbf{f}(\boldsymbol{\eta}_A, \boldsymbol{\mu}_A, \mathbf{a}_A) \tag{4.12}$$

and

$$\mathbf{x}_B = \mathbf{f}(\boldsymbol{\eta}_B, \boldsymbol{\mu}_B, \mathbf{a}_B) \tag{4.13}$$

The given model of robot A provides for the functional structure $\mathbf{f}(\cdot)$ that is assumed to be the same in robot B as well. The identification problem is thus reduced to a parameter identification problem. Specifically, the model of robot A provides for the nominal values of the kinematic parameters

$$\boldsymbol{\mu}_A = \boldsymbol{\mu}_0 \tag{4.14}$$

and

$$\mathbf{a}_A = \mathbf{a}_0 \tag{4.15}$$

The vectors $\boldsymbol{\mu}_B$ and $\mathbf{a}_B$ are unknown. Estimates $\hat{\boldsymbol{\mu}}_B$ and $\hat{\mathbf{a}}_B$ are to be constructed based on a set of measurement data collected from robot B. That is, the end effector of robot B is placed at m locations, $[\mathbf{x}_B(1), \ldots, \mathbf{x}_B(m)]$, within the robot workspace. For each of the m robot configurations, the relationship between the workspace position and the joint displacement transducers will be given by

$$\mathbf{x}_B(j) = \mathbf{f}[\boldsymbol{\eta}_B(j), \boldsymbol{\mu}_B, \mathbf{a}_B] \qquad j = 1, \ldots, m \tag{4.16}$$

End point sensing methods are used to determine some of the elements of each vector $\mathbf{x}_B(j), j = 1, \ldots, m$. Also at each such configuration the vector $\boldsymbol{\mu}_B(j)$ is read.

Let the total number of unknown parameters be denoted by N_p:

$$N_p = \dim(\boldsymbol{\mu}_B) + \dim(\mathbf{a}_B) \tag{4.17}$$

If there is no redundancy in the sets of parameters $\boldsymbol{\mu}$ and $\mathbf{a}$ then N_p is a necessary but not sufficient lower bound on the minimum number of scalar algebraic equations that need to be specified by the measurements. Neither $\boldsymbol{\mu}_B$ nor $\mathbf{a}_B$ is really constant. Both may vary to properly model the effect of nongeometric terms. Many model based parameter identification methods, though, simply ignore such variations and treat the unknowns as constants.

Some calibration measurement techniques are such that at each robot configuration the entire 6-vector $\mathbf{x}_B(j)$ is measured or can be calculated from measured entities. On the other hand certain measurement techniques provide only for a subset of the full cartesian position vector. In the latter case, Equation 4.16 needs to be modified as follows: Let the vector $\mathbf{y}_B(j)$ represent those elements of $\mathbf{x}_B(j)$ that are either directly measurable by the particular measurement technique that is being used, or can be determined from the constraint equations imposed by the measurement action. Then

$$\mathbf{y}_B(j) = \mathbf{f}_y[\boldsymbol{\eta}_B(j), \boldsymbol{\mu}_B, \mathbf{a}_B] \qquad j = 1, \ldots, m \tag{4.18}$$

where $\mathbf{f}_y(\cdot)$ represents the appropriate subset of kinematic equations.

In practice the vector $\mathbf{y}_B(j)$ cannot be measured without error. The measurement noise $\mathbf{v}(j)$ obviously depends on the accuracy and resolution of the end

point sensors and fabrication tolerances of the calibration fixtures

$$\mathbf{v}(j) = \mathbf{y}_B(j) - \mathbf{y}_{B,m}(j) \qquad j = 1, \ldots, m \tag{4.19}$$

where $\mathbf{y}_{B,m}(j)$ is the position that is actually read by the end-point sensors, or calculated from the end-point sensing data. In the latter, if $\mathbf{y}_{B,m}(j)$ is found through arithmetic manipulations, numerical errors of the processing method also contribute to $\mathbf{v}(j)$.

The reading of the joint transducer $\boldsymbol{\eta}_B(j)$ may also include errors. The most obvious errors are due to quantization noise inherent to any digital conversion of analog data. Other types of errors may be modeled as noise to "mask" nonlinearities or fluctuations in the relationship between the output of the joint sensor and the actual joint displacement. To elaborate on this point, recall η_i the signal coming from a given joint position transducer, and the actual joint displacement q_i. Ideally the relationship between q_i and η_i is linear as follows:

$$q_i = k_{i1}\eta_i + k_{i2} \tag{4.20}$$

In such a case one denotes

$$\boldsymbol{\mu}_i = (k_{i1}, k_{i2})^T \tag{4.21}$$

The vector $\boldsymbol{\mu}$ is the concatenation of all such pairs $\boldsymbol{\mu}_i$. The physical significance of this vector depends on the particular types of transducer or drive systems and relative location of the transducer with respect to the joint axis of motion. For example, if an incremental encoder is being used, k_{i1} will represent the encoder gain including any gain added by gearing of the encoder shaft to the joint. The constant k_{i2} represents the displacement of the joint when the encoder counter is set to zero.

In cases involving high precision it may become necessary to develop a more sophisticated model than the one in Equation 4.20. Nonlinear effects such as gear backlash or transducer nonlinearities may be included in the model. For example, Whitney et al. [31] in their experiments with a Puma 560 robot detected a harmonic variation superimposed on the linear relationship. This effect, which was mostly attributed to gear eccentricity, was empirically modeled as

$$q_i = k_{i1}\eta_i + k_{i2} + k_{i3}\sin(k_{i1}\eta_i + k_{i4}) \tag{4.22}$$

where the constants k_{i3} and k_{i4} described the magnitude of the harmonic variation and the phase shift. In some cases the design of the joint and type of transducer will dictate the form of the model used. In many instances, however, the particular form of the nonlinearities must be deduced from experimental data. Alternatively, one may choose to model any additive nonlinear term such as the one appearing in Equation 4.22 as an additive noise ζ_i that "masks" the nonlinearities in the drive model.

The total measurement data Ω consists of m pairs of vectors as follows:

$$\Omega = \{[\mathbf{y}_{B,m}(1), \boldsymbol{\eta}_B(1)], \ldots, [\mathbf{y}_{B,m}(m), \boldsymbol{\eta}_B(m)]\} \tag{4.23}$$

The parameter identification algorithm maps Ω and the a priori data (i.e., nominal values of kinematic parameters and possibly probabilistic data regarding the measurement noise and the unknown parameters) into a unique estimate of the vectors $\boldsymbol{\mu}_B$ and $\mathbf{a}_B$.

4.3 LINEAR LEAST-SQUARES PARAMETER ESTIMATION

This section is an introduction to standard least-squares estimation, minimum variance estimation, and Kalman filters as they apply to the problem of estimating the errors in the robot kinematic parameters. The emphasis is not so much on the algorithmic side but more on estimation error analysis. We will begin by casting the robot calibration equations in a framework more compatible with these estimation techniques.

Referring to the vector formulation of the robot kinematics given in Equation 4.11, let $\mathbf{X}$ be defined as the vector of changes in the kinematic parameters between robot A and robot B:

$$\mathbf{X} = (\Delta\boldsymbol{\mu}^T, \Delta\mathbf{a}^T)^T \tag{4.24}$$

where

$$\Delta\mathbf{a} = \mathbf{a}_B - \mathbf{a}_A \tag{4.25}$$

$$\Delta\boldsymbol{\mu} = \boldsymbol{\mu}_B - \boldsymbol{\mu}_A \tag{4.26}$$

Robot A is the nominal robot and robot B is the actual robot.

The identification of the kinematic model of robot B is the problem of estimating the vector $\mathbf{X}$ based on the collected observations.

The convergence of many linear identification algorithms may depend on the "smallness" of $\mathbf{X}$ in a certain sense.

The small perturbation requirement (i.e., small $\|\mathbf{X}\|$) may be necessary to establish a linearized measurement equation. That is, an equation in which the vector that depends on observed data is a linear function of the "signal" vector of unknowns $\mathbf{X}$ as was shown in the modeling chapter.

To find the relationship between a vector of small perturbations $\mathbf{X}$ and the measurements, the following residual or "world coordinate error vector" $\mathbf{e}(j)$ is defined:

$$\mathbf{e}(j) = \mathbf{f}_y[\boldsymbol{\eta}_B(j), \boldsymbol{\mu}_B, \mathbf{a}_B] - \mathbf{f}_y[\boldsymbol{\eta}_B(j), \boldsymbol{\mu}_A, \mathbf{a}_A] \tag{4.27}$$

where $\mathbf{f}_y(\cdot)$ is as in Equation 4.18. The vector $\mathbf{e}(j)$ is the difference between $\mathbf{x}_B(j)$

and $\mathbf{x}_A(j)$ (the robot pose vectors), for the same readings of the joint transducers. By neglecting higher order powers of the components of $\mathbf{X}$ and cross-products of elements of $\mathbf{X}$, it may be shown that $\mathbf{e}(j)$ relates linearly to $\mathbf{X}$:

$$\mathbf{e}(j) \simeq \mathbf{H}(j)\mathbf{X} \tag{4.28}$$

where the matrix $\mathbf{H}(j)$ depends on the particular robot configuration as determined by the vector of joint readings $\boldsymbol{\eta}_B(j)$.

Combining Equations 4.19, 4.27, and 4.28 yields

$$\mathbf{H}(j)\mathbf{X} = \mathbf{y}_{B,m}(j) - \mathbf{f}_y[\boldsymbol{\eta}_B(j), \boldsymbol{\mu}_A, \mathbf{a}_A] + \mathbf{v}_B(j) \tag{4.29}$$

A generalized "measurement vector" $\mathbf{z}(j)$ is now defined as follows:

$$\mathbf{z}(j) = \mathbf{y}_{B,m}(j) - \mathbf{f}_y[\boldsymbol{\eta}_B(j), \boldsymbol{\mu}_A, \mathbf{a}_A] \tag{4.30}$$

where $\mathbf{z}(j)$ is the computed difference between the observation vector $\mathbf{y}_{B,m}(j)$, measured using end point sensors, and a computed observation vector using the nominal kinematic model and the measured joint positions. Combining Equations 4.29 and 4.30 results in

$$\mathbf{z}(j) = \mathbf{H}(j)\mathbf{X} - \mathbf{v}_B(j) \qquad j = 1, \ldots, m \tag{4.31}$$

This linearized measurement equation is the starting point to a variety of linear identification techniques including the least-squares estimation $\hat{\mathbf{X}}$ of $\mathbf{X}$ and minimum-variance estimate of $\mathbf{X}$.

4.3.1 Standard Linear Least-Squares Estimation

The linearized error model is the basis for direct linear identification methods of the errors in the kinematic parameters. Again, let the vector $\mathbf{X}$ denote the unknown errors in the robot kinematic parameters. During the measurement phase of calibration, the robot is brought to m different configurations. In each measurement configuration, a measurement vector $\mathbf{z}(j)$ is determined. The vector $\mathbf{z}(j)$ is the difference between the measured end position and the end point position as calculated from the nominal robot kinematic model. At each measurement pose an unavoidable measurement error vector $\mathbf{v}(j)$ exists. The linearized measurement equation can be expressed as follows:

$$\mathbf{z}(j) = \mathbf{H}(j)\mathbf{X} + \mathbf{v}(j) \qquad j = 1, \ldots, m \tag{4.32}$$

where the matrix $\mathbf{H}(j)$ is defined from the robot *nominal* kinematic model at each measurement configuration $j, j = 1, \ldots, m$. The dimension of $\mathbf{z}(j)$ is 6 or less and will be denoted by l.

In the context of recursive estimation, Equation 4.32 may be viewed as a "discrete-time" measurement equation. The time index j represents of course a completely arbitrary ordering of the robot measurement configurations, $j = 1, \ldots, m$. Sometimes it may be convenient to assume that $j = 0$ corresponds to the robot home position. As the matrix $\mathbf{H}(j)$ changes from one measurement configuration to another, Equation 4.32 is essentially a "time-varying" measurement equation. However, unlike standard time-varying system models, here the time index j does not appear explicitly in the elements of the matrix $\mathbf{H}(\cdot)$. Instead, the numerical values of $\mathbf{H}(\cdot)$ are known at every value of j.

A batch-processing strategy is to concatenate all measurements $\mathbf{z}(j)$ into a single measurement vector $\mathbf{Z}$:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}(1) \\ \vdots \\ \mathbf{z}(m) \end{bmatrix} \tag{4.33}$$

The m equations given in Equation 4.32 may now be written as

$$\mathbf{Z} = \mathbf{H}\mathbf{X} + \mathbf{V} \tag{4.34}$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}(1) \\ \vdots \\ \mathbf{H}(m) \end{bmatrix} \tag{4.35}$$

and

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}(1) \\ \vdots \\ \mathbf{v}(m) \end{bmatrix} \tag{4.36}$$

When solving a set of linear algebraic equations in which the free terms in each equation depend on measured data that may contain error, it makes sense to take enough measured data so that the number of equations be greater than the number of unknowns. This is done to reduce the influence of errors on the computation results. Thus, $\mathbf{X}$, the vector of unknowns is n-dimensional. $\mathbf{Z}$, the vector of measured data is p-dimensional where $p = ml$. It is assumed that $p \geq n$. In many practical applications $p \gg n$ is common. Also note that $\mathbf{V}$ is p-dimensional as well.

In the standard least-squares method, no stochastic modeling is done. In other words, the noise vector $\mathbf{V}$ is treated as an unknown variable.

The goal is to find an estimate $\hat{\mathbf{X}}$ of the unknown $\mathbf{X}$. In particular one tries to find $\hat{\mathbf{X}}$ that minimizes the sum of squares of the elements of the errors vector $\mathbf{Z} - \mathbf{H}\hat{\mathbf{X}}$.

For that purpose a scalar cost function J is created:

$$J = (\mathbf{Z} - \mathbf{H}\hat{\mathbf{X}})^T(\mathbf{Z} - \mathbf{H}\hat{\mathbf{X}}) \tag{4.37}$$

This performance criterion is quadratic in the elements of $\hat{\mathbf{X}}$. The minimum value of J is obtained when

$$\frac{\partial J}{\partial \hat{\mathbf{X}}} = 0 \tag{4.38}$$

together with the condition that at the value $\hat{\mathbf{X}}$ that satisfies Equation 4.38, the Hessian matrix of J is positive semidefinite:

$$\frac{\partial^2 J}{\partial \hat{\mathbf{X}}^2} \geq 0 \tag{4.39}$$

To differentiate J, the following useful matrix derivative formulas are used:

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{y}) = \mathbf{A}\mathbf{y} \tag{4.40}$$

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{y}^T\mathbf{A}\mathbf{x}) = \mathbf{A}^T\mathbf{y} \tag{4.41}$$

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = (\mathbf{A} + \mathbf{A}^T)\mathbf{x} \tag{4.42}$$

where $\mathbf{A}$ is any square matrix and $\mathbf{x}$ and $\mathbf{y}$ are vectors of appropriate dimensions.

Differentiating Equation 4.37 and substituting into the condition given in Equation 4.38 yields

$$\mathbf{H}^T\mathbf{H}\hat{\mathbf{X}} = \mathbf{H}^T\mathbf{Z} \tag{4.43}$$

It can be shown that under Equation 4.43, the second derivative of J with respect to $\hat{\mathbf{X}}$ is positive semidefinite. Thus, Equation 4.43 indeed defines a minimum.

When the matrix $\mathbf{H}^T\mathbf{H}$ is nonsingular, the least-squares estimate is

$$\hat{\mathbf{X}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Z} \tag{4.44}$$

The performance criterion in Equation 4.37 gives equal weighting to all the elements of the error vector. In robotic applications there are particular kinematic errors that may be more important than others. For instance, it was

demonstrated in [32] that errors in the rotational θ and α DH parameters dominated errors in the translational parameters in their effect on the overall robot accuracy. To provide for such unequally weighted estimation the performance measure J may be modified as follows:

$$J = (\mathbf{Z} - \mathbf{H}\hat{\mathbf{X}})^T\mathbf{W}(\mathbf{Z} - \mathbf{H}\hat{\mathbf{X}}) \tag{4.45}$$

where the weighting matrix $\mathbf{W}$ is taken to be symmetric and positive definite. The minimizing solution now satisfies the equation

$$\mathbf{H}^T\mathbf{W}\mathbf{H}\hat{\mathbf{X}} = \mathbf{H}^T\mathbf{W}\mathbf{Z} \tag{4.46}$$

Again, a condition of nonzero determinant is posed on the matrix $\mathbf{H}^T\mathbf{W}\mathbf{H}$, and then the least-squares estimator is

$$\hat{\mathbf{X}} = (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}\mathbf{Z} \tag{4.47}$$

The estimator coefficients depend only on $\mathbf{H}$, the deterministic part of the measurement model and on the weighting matrix $\mathbf{W}$.

The estimator has been derived using deterministic reasoning only. It is of interest though to study some of the probabilistic properties of $\hat{\mathbf{X}}$ in Equation 4.44 or 4.47.

Let $\mathbf{e}$ denote the estimation error vector:

$$\mathbf{e} = \mathbf{X} - \hat{\mathbf{X}} \tag{4.48}$$

By Equations 4.34 and 4.47:

$$\mathbf{e} = \mathbf{X} - (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}(\mathbf{H}\mathbf{X} + \mathbf{V}) = -(\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}\mathbf{V} \tag{4.49}$$

The expected value of the error $E(\mathbf{e})$ is thus

$$\mathbf{E}(\mathbf{e}) = -(\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}E(\mathbf{V}) \tag{4.50}$$

If all we know about the measurement noise $\mathbf{V}$ is that it is a zero-mean random vector, then the estimation error using a linear least-squares estimator is also zero mean. An estimator with such property is called an "unbiased" estimator.

Proceeding one further step, it is assumed that the measurement noise in addition to having a zero mean also has the following noise covariance matrix:

$$\mathrm{Var}(\mathbf{V}) \equiv \mathbf{\Sigma}_v \equiv E\{(\mathbf{V} - E(\mathbf{V}))(\mathbf{V} - E(\mathbf{V}))^T\} = E(\mathbf{V}\mathbf{V}^T) \tag{4.51}$$

where the matrix $\mathbf{\Sigma}_v$ is symmetric and positive definite. $\mathrm{Var}(\mathbf{V})$ describes the noise "intensity." If absolutely no information is known about the measurement error size, and the error elements may be arbitrarily large, then the following is taken

$$\Sigma_v^{-1} = 0 \tag{4.52}$$

As demonstrated in [20], the estimation error covariance Σ_e

$$\text{Var}(\mathbf{e}) = \Sigma_e = E(\mathbf{e}\mathbf{e}^T) \tag{4.53}$$

can now be expressed in terms of the measurement noise covariance, using Equation 4.48, as follows:

$$\begin{aligned}\Sigma_e &= E\{(\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}\mathbf{V}\mathbf{V}^T\mathbf{W}^T\mathbf{H}(\mathbf{H}^T\mathbf{W}^T\mathbf{H})^{-1}\} \\ &= (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}\Sigma_v\mathbf{W}\mathbf{H}(\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\end{aligned} \tag{4.54}$$

Finally, note that singular cases, either $\mathbf{H}^T\mathbf{H}$ or $\mathbf{H}^T\mathbf{W}\mathbf{H}$ being singular matrices, indicate an ill-conditioned measurement model as given by Equation 4.34. This is the case in which some of the linear algebraic equations in the set of measurement equations are linearly dependent on other equations. The measurement model should then be modified by removing all such redundant equations.

4.3.2 Linear Minimum Variance Estimation

In the previous section the mean and variance of the least-squares estimation error have been studied under the assumption that limited probabilistic information was made available about the measurement noise $\mathbf{V}$, namely the first two moments of $\mathbf{V}$. A question may now be asked as to whether the least-squares estimator is optimal also in the sense of giving minimum error variance. In general the answer to this question is no.

In this section it is assumed that the first- and second-order moments of the measurement noise $\mathbf{V}$ and the unknown vector $\mathbf{X}$ are all given as follows:

$$E(\mathbf{X}) = \mu_x \tag{4.55}$$

$$\text{Var}(\mathbf{X}) = \Sigma_x \tag{4.56}$$

$$E(\mathbf{V}) = 0 \tag{4.57}$$

$$\text{Var}(\mathbf{V}) = \Sigma_v \tag{4.58}$$

Where the covariance matrices Σ_x and Σ_v are both symmetric and positive definite. It is further assumed that $\mathbf{V}$ and $\mathbf{X}$ are uncorrelated.

An estimator of a particular structure is of interest. That is an estimator in which the measurement vector $\mathbf{Z}$ is processed linearly. Such a linear estimator has the following structure:

$$\hat{\mathbf{X}} = \mathbf{b} + \mathbf{A}\mathbf{Z} \tag{4.59}$$

The objective is to select an n-vector $\mathbf{b}$ and a $n \times p$ matrix $\mathbf{A}$ to minimize the

estimation error variance. It should be noted though that such an optimal linear estimator is not necessarily the optimal minimum variance estimator. In general, there may exist a nonlinear estimator $\hat{\mathbf{X}} = f(\mathbf{Z})$ that provides an even smaller error variance.

As it turns out, the design parameters **b** and **A** provide more design parameters than are required for the optimization problem. Thus, one can afford imposing additional requirements on the estimator. Namely, the estimator should also be unbiased in the following sense:

$$E(\hat{\mathbf{X}}) = E(\mathbf{X}) \tag{4.60}$$

Therefore, using Equation 4.55, 4.59, an 4.34.

$$\mathbf{b} + \mathbf{A}\mathbf{H}\boldsymbol{\mu}_x = \boldsymbol{\mu}_x \tag{4.61}$$

Thus

$$\hat{\mathbf{X}} = \boldsymbol{\mu}_x + \mathbf{A}(\mathbf{Z} - \mathbf{H}\boldsymbol{\mu}_x) \tag{4.62}$$

This structure guarantees that the estimation error is zero mean. The matrix **A** needs to be selected to minimize the error variance. As the error variance is a matrix and since the performance measure should be a scalar, a common approach is to minimize the sum of the variances of each component of the error vector. This is the sum of main diagonal terms of the error covariance matrix known as the trace of the error covariance matrix:

$$J = \mathrm{tr}\{\mathrm{Var}(\mathbf{X} - \hat{\mathbf{X}})\} = \mathrm{tr}(\boldsymbol{\Sigma}_e) \tag{4.63}$$

J needs to be minimized by searching over all $n \times p$ matrices **A**:

$$J_{\text{optimal}} = \min_{\mathbf{A}} \mathrm{tr}\{E[(\mathbf{X} - \boldsymbol{\mu}_x - \mathbf{A}(\mathbf{Z} - \mathbf{H}\boldsymbol{\mu}_x))(\mathbf{X} - \boldsymbol{\mu}_x - \mathbf{A}(\mathbf{Z} - \mathbf{H}\boldsymbol{\mu}_x))^T]\} \tag{4.64}$$

Denoting by $\mathbf{A}^*$ the optimal **A** that minimizes J of Equation 4.63 the result, using standard calculus of variations as given in [20] is

$$\mathbf{A}^* = \boldsymbol{\Sigma}_x\mathbf{H}^T(\mathbf{H}\boldsymbol{\Sigma}_x\mathbf{H}^T + \boldsymbol{\Sigma}_v)^{-1} \tag{4.65}$$

yielding the linear minimum variance estimator

$$\hat{\mathbf{X}} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_x\mathbf{H}^T(\mathbf{H}\boldsymbol{\Sigma}_x\mathbf{H}^T + \boldsymbol{\Sigma}_v)^{-1}(\mathbf{Z} - \mathbf{H}\boldsymbol{\mu}_x) \tag{4.66}$$

It is noticed that in the estimator formula (Equation 4.66) the matrix that needs to be inverted is of size $p \times p$ where p is the dimension of the measurement vector **Z**. Equation 4.66 may be simplified in the sense that the matrix to be inverted will be of size $n \times n$ instead of $p \times p$ where n is the dimension of the unknown vector **X**.

This involves the well known matrix inversion identity [20] as follows.

Let **P** and **R** be nonsingular matrices of size $n \times n$ and $p \times p$, repectively, and H be a $p \times n$ matrix. Then

$$(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} + \mathbf{P}^{-1})^{-1} = \mathbf{P} - \mathbf{P}\mathbf{H}^T(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\mathbf{P} \tag{4.67}$$

As an application of this identity let

$$\mathbf{P} = \boldsymbol{\Sigma}_x \qquad \mathbf{R} = \boldsymbol{\Sigma}_v \tag{4.68}$$

Then Equation 4.66 may be written as:

$$\hat{\mathbf{X}} = (\mathbf{H}^T\boldsymbol{\Sigma}_v^{-1}\mathbf{H} + \boldsymbol{\Sigma}_x^{-1})^{-1}(\mathbf{H}^T\boldsymbol{\Sigma}_v^{-1}\mathbf{Z} + \boldsymbol{\Sigma}_x^{-1}\boldsymbol{\mu}_x) \tag{4.69}$$

which is easier to implement than Equation 4.66 since in practice $n \ll p$. Furthermore, the effects of complete ignorance of probabilistic models may be easily incorporated into Equation 4.69. If the variance of **X** is arbitrarily large, that can be modeled by setting

$$\boldsymbol{\Sigma}_x^{-1} = 0$$

Similarly, if the measurement noise is arbitrarily large, one should substitute

$$\boldsymbol{\Sigma}_v^{-1} = 0$$

into Equation 4.69. In other words, if the measurements are so noisy that the signal cannot be recognized, the minimum-variance estimation strategy would be simply to take

$$\hat{\mathbf{X}} = \boldsymbol{\mu}_x$$

It is now a straightforward exercise to compute the estimation error covariance of $\hat{\mathbf{X}}$ of Equation 4.69. The important result is

$$\boldsymbol{\Sigma}_e = (\mathbf{H}^T\boldsymbol{\Sigma}_v^{-1}\mathbf{H} + \boldsymbol{\Sigma}_x^{-1})^{-1} \tag{4.70}$$

Equation 4.70 relates the estimation error to the intensity of the measurement noise **V** and to the amount of uncertainty in the unknown vector **X**.

At this point it is of interest to compare the least-squares estimator to the linear minimum variance estimator. Since the least-squares estimator is also a linear estimator, its performance in terms of error variance is in general inferior compared to the optimal estimator derived in this section.

By comparing Equations 4.54 and 4.70, it is observed that the least-squares estimator can be made a minimum variance estimator if one chooses the weighting matrix **W** to be

$$\mathbf{W} = \boldsymbol{\Sigma}_v^{-1} \tag{4.71}$$

In other words, a possible strategy for least-squares estimation is to give weighting according to the noise intensity of every measurement configuration. The "noisier" the measurement, the smaller is its effect on the estimation.

So far, it has been assumed that the probability distribution functions of the random vectors **V** and **X** are not fully known. Estimators have been developed in cases where the first two moments of these distributions have been given. A common approach to stochastic modeling of many physical systems is to assume that each of the random vectors **V** and **X** is Gaussian. As is well known, the Gaussian probability distribution function is fully characterized in terms of its first two moments. Therefore, the previously developed estimators may be shown to be optimal with respect to some additional performance criteria under the assumption of Gaussian measurement noise and Gaussian unknown vector **X**.

For instance, a "maximum-likelihood estimation" philosophy is to choose **X** such that it maximizes the probability of measurements **Z** that actually occurred. Assuming that no information is available about **X** and that **V** is zero mean, Gaussian with covariance matrix **R**, the conditional probability density of **Z**, given **X**, is:

$$\mathbf{p}(\mathrm{Z}|\mathrm{X}) = \frac{1}{(2\pi)^{p/2}|\mathbf{R}|^{1/2}} \exp[-\tfrac{1}{2}(\mathbf{Z} - \mathbf{HX})^T \mathbf{R}^{-1}(\mathbf{Z} - \mathbf{HX})] \tag{4.72}$$

where p is the dimension of **Z**. Choosing a vector $\hat{\mathbf{X}}$ that maximizes $p(\mathbf{Z}|\mathbf{X})$ is now equivalent to minimizing the cost function

$$J = (\mathbf{Z} - \mathbf{H}\hat{\mathbf{X}})^T \mathbf{R}^{-1}(\mathbf{Z} - \mathbf{H}\hat{\mathbf{X}}) \tag{4.73}$$

Thus, the standard least squares estimator for a weighting matrix $\mathbf{W} = \mathbf{R}^{-1}$ is also the optimal maximum likelihood estimator under the further assumption of a Gaussian measurement noise.

One may also show that under the assumptions of Gaussian **V** and **X**, the linear minimum-variance estimator for the linear measurement model is also the optimal minimum-variance estimator among all possible nonlinear estimators.

4.3.3 Linear Least-Squares—Practical Considerations

The International Mathematical and Statistical Library (IMSL) [14] contains a routine LSQRR that solves a linear least-squares problem with iterative refinement. This routine is also available in a double precision version, DLSQRR. Given a linear system of equations of the form $\mathbf{b} = \mathbf{Ax}$ where **x** is $n \times 1$, **b** is $m \times 1$, **A** is $m \times n$, and m is greater than n, the routine computes **x** from **A** and **b** using a least-squares fit similar to that in Equation 4.44. The IMSL routines are powerful, well debugged, and easy to use. There are some programming details that need explanation, however, and these are discussed in the following sections. The argument TOL in the call to LSQRR is used to determine the subset of columns of **A** to be included in the solution. One wishes, of course, to use all

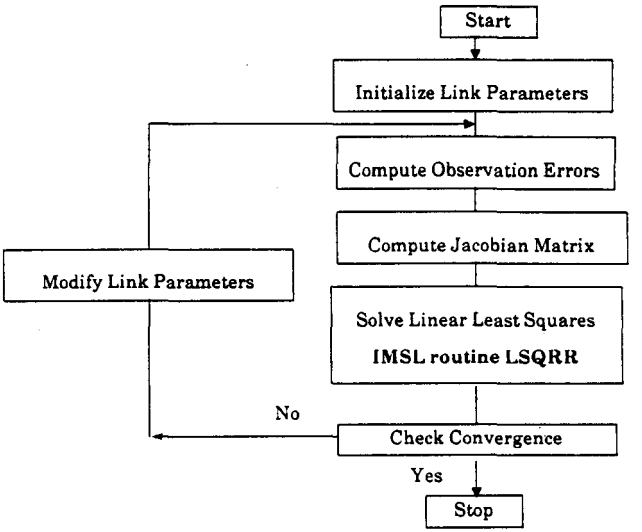


Figure 4.1. Flowchart for linear least squares.

columns of the identification Jacobian, hence, TOL is set to zero. Note that it may be necessary to scale the rows and columns of the **A** matrix as explained in the IMSL documentation. In particular, as the elments of **b** are subject to random errors, the equations should be scaled so that the variance in b_i, $i = 1, \ldots, m$ is constant. Thus, six degree of freedom pose measurement for calibration would imply that the rows of the Jacobian corresponding to position error and the rows corresponding to orientation would be scaled differently, depending on the units used.

The flow chart for the linear least-squares method is shown in Figure 4.1. The parameter **k** as defined in Equation 4.2 is initially set to the nominal manipulator link parameter values. Using **k** and the vector of joint variables, **q**, for each observation, the predicted manipulator transformation, $\mathbf{T}_p$, is calculated. The observation errors are calculated from the predicted and measured end effector pose. The elements of the Jacobian matrix are calculated next. The linear least-squares routine LSQRR is then invoked to compute the parameter error vector, $\delta\mathbf{k}$ from **J** and $\delta\mathbf{T}$ as defined in Equation 4.6. Since the calculation of the identification Jacobian assumes linear approximations, the parameter errors are added back to the parameter vector as

$$\mathbf{k}_{\text{new}} = \mathbf{k}_{\text{old}} + \delta\mathbf{k} \tag{4.74}$$

and the linear least-squares procedure is repeated. To check for a stopping criterion, the sum of squares of the identified parameter errors is computed at the end of each iteration. If the identification is unsuccessful, this sum diverges to larger and larger values. It should be noted that neither convergence nor

divergence is necessarily "monotonic." The LSQRR routine returns the residual vector ($\mathbf{b} - \mathbf{Ax}$) in the argument RES, and the scalar containing the estimated basis of the coefficient matrix $\mathbf{A}$ in the argument KBASIS. KBASIS is thus also the rank of the Jacobian and its value is an indication of model singularity when it is less the number of parameters to be identified.

The reader is referred to the simulation example in Section 4.5 that illustrates actual results of running the least squares routine described above.

4.3.4 Kalman Filtering

The estimation techniques that have been discussed so far are all "batch processing" algorithms. In other words, all the data are first collected and then the processing is done on the entire set of measurements simultaneously. An alternative approach is recursive estimation. In a recursive estimator the measurements come sequentially and there is no need to store past measurements for the purpose of computing present estimates. Whereas in batch estimation the size of the measurement vector must be larger than the size of the estimated vector, one typically experiences the opposite in recursive estimation.

In this section, the key properties of the discrete time Kalman filter as applied to robot calibration are discussed. The derivation of the various filter formulas follow those given in reference [8,20]. The Kalman filtering problem is the problem of finding an optimal state estimation in a linear Gaussian time varying dynamic system. In other words, rather than assuming a fixed unknown vector $\mathbf{X}$, a vector that evolves in time according to a linear difference equation is assumed as follows

$$\mathbf{X}_k = \mathbf{A}_{k-1}\mathbf{X}_{k-1} + \mathbf{w}_{k-1} \tag{4.75}$$

where $\mathbf{X}_k$ is an n-vector called the "state vector." The state at time $t = t_k$ is denoted by $\mathbf{X}_k$. $\mathbf{A}_{k-1}$ is an $n \times n$ matrix whose elements may in general depend on the time index k. The vector $\mathbf{w}_k$ is a zero mean white noise sequence of covariance $\mathbf{Q}_k$. It is assumed that initially, at $t = 0$, the state $\mathbf{X}_0$ is a Gaussian random vector that is independent of any of the vectors $\mathbf{w}_k$, $k = 0, 1, 2, \ldots$. Equation 4.75 is convenient whenever one wants to model imperfect joints. The vector $\mathbf{X}$ of kinematic parameter errors may in practice fluctuate randomly as the robot joint axes "wobble."

The measurement vector at time $t = t_k$, denoted as $\mathbf{Z}_k$, is assumed to be a linear combination of the state variables at that time, corrupted by a measurement noise. The measurement model is as follows:

$$\mathbf{Z}_k = \mathbf{H}_k\mathbf{X}_k + \mathbf{V}_k \tag{4.76}$$

where the dimension of $\mathbf{Z}_k$ is l (typically $l \leq n$). $\mathbf{H}_k$ is an $l \times n$ time-varying matrix and $\mathbf{V}_k$ denotes a zero mean white noise sequence of covariance $\mathbf{R}_k$. It is assumed that the measurement noise sequence $\mathbf{V}_k$ is statistically independent of both the process noise sequence $\mathbf{w}_k$ and the initial state $\mathbf{X}_0$.

The objective is to find a recursive state estimate $\hat{\mathbf{X}}_k$ that is a linear combination of the present measurement $\mathbf{Z}_k$ and the previous state estimate that depends on all the past measurements $\mathbf{Z}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_{k-1}$. In addition, $\hat{\mathbf{X}}_k$, needs to be optimal in a certain sense.

For simplicity let the performance objective be the minimum-variance of the estimation error. In other words, $\hat{\mathbf{X}}_k$ will be derived here to be the linear minimum-variance estimator. What will not be shown here is that under the assumptions that both the process $\mathbf{X}_k$ is linear and Gaussian and the measurement noise $\mathbf{V}_k$ is Gaussian, the same $\hat{\mathbf{X}}_k$ may be shown to be optimal with respect to many other performance criteria as well including nonlinear minimum-variance estimation, maximum likelihood estimation, and more.

To mathematically quantify the recursiveness requirement, the following notation will be used. The term $\hat{\mathbf{X}}_k(-)$ represents the estimate of the system state at time t_k without taking into account the measurement $\mathbf{Z}_k$. The term $\hat{\mathbf{X}}_k(+)$ represents the updated version of $\hat{\mathbf{X}}_k(-)$ using the measurement $\mathbf{Z}_k$. An estimate of the following functional structure is then sought:

$$\hat{\mathbf{X}}_k(+) = \mathbf{C}_k\hat{\mathbf{X}}_k(-) + \mathbf{K}_k\mathbf{Z}_k \tag{4.77}$$

where the time-varying matrices $\mathbf{C}_k$ and $\mathbf{K}_k$ are as yet unspecified. For simplicity, we will set

$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_k(+) \tag{4.78}$$

To understand the reason why $\hat{\mathbf{X}}_k$ is *not* taken to be the linear combination of $\hat{\mathbf{X}}_{k-1}$ and $\mathbf{Z}_k$, note that the measurements $\mathbf{Z}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_k$ come only at discrete times $0, t_1, t_2, \ldots, t_k$. Since no information is coming in between sampling instances, the relationship between $\hat{\mathbf{X}}_k(-)$ and $\hat{\mathbf{X}}_{k-1}$ is determined by the way the state propagates between t_{k-1} and t_k according to Equation 4.75 without taking into account the process noise.

$$\hat{\mathbf{X}}_k(-) = \mathbf{A}_{k-1}\hat{\mathbf{X}}_{k-1} = \mathbf{A}_{k-1}\hat{\mathbf{X}}_{k-1}(+) \tag{4.79}$$

Obviously, $\hat{\mathbf{X}}_k(-)$ represents the most updated state estimate up to the moment t_k when a new measurement is about to be added.

Requiring that the estimator be unbiased for every k, that is

$$E(\mathbf{e}_k) = E(\mathbf{X}_k - \hat{\mathbf{X}}_k) = 0 \tag{4.80}$$

imposes additional restrictions on the choice of the matrices $\mathbf{C}_k$ and $\mathbf{K}_k$ of Equation 4.77. To simplify, we make the following definitions:

$$\mathbf{e}_k = \mathbf{X}_k - \hat{\mathbf{X}}_k \tag{4.81}$$

$$\mathbf{e}_k(-) = \mathbf{X}_k - \hat{\mathbf{X}}_k(-) \tag{4.82}$$

Obviously, if $E(\mathbf{e}_k) = 0$ for any k, then $E[\mathbf{e}_k(-)] = 0$ also. Now, by substituting Equations 4.76 and 4.77 into $\mathbf{e}_k$, one gets

$$\mathbf{e}_k = (\mathbf{I} - \mathbf{C}_k - \mathbf{K}_k\mathbf{H}_k)\mathbf{X}_k + \mathbf{C}_k\mathbf{e}_k(-) - \mathbf{K}_k\mathbf{V}_k \tag{4.83}$$

Then the requirement that the estimator be unbiased implies that

$$\mathbf{C}_k = \mathbf{I} - \mathbf{K}_k\mathbf{H}_k \tag{4.84}$$

The unbiased estimator takes the form

$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_k(-) + \mathbf{K}_k[\mathbf{Z}_k - \mathbf{H}_k\hat{\mathbf{X}}_k(-)] \tag{4.85}$$

and the estimation error is

$$\mathbf{e}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{e}_k(-) + \mathbf{K}_k\mathbf{V}_k \tag{4.86}$$

Examining closely expressions 4.79 and 4.85 one notices that any arbitrary estimator using an arbitrary gain matrix $\mathbf{K}_k$ has the simple block diagram shown in Figure 4.2. The estimator consists of an internal model that duplicates the known model of the process and measurements $\mathbf{A}_k$, $\mathbf{H}_k$. The vector $\mathbf{v}_k$

$$\mathbf{v}_k \triangleq \mathbf{Z} - \mathbf{H}_k\hat{\mathbf{X}}_k(-) \tag{4.87}$$

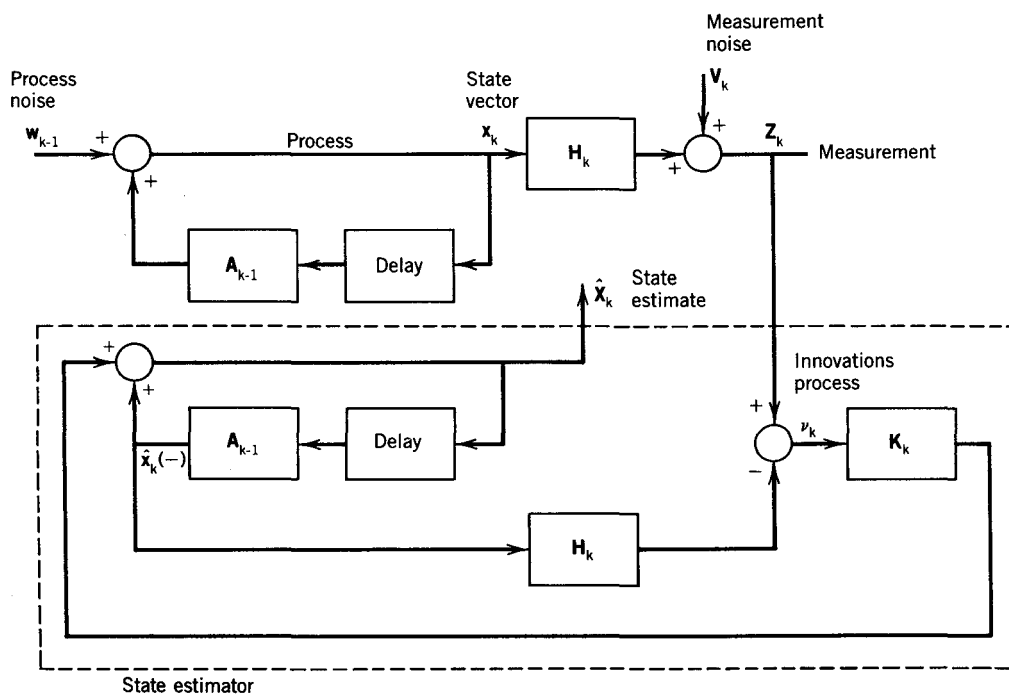


Figure 4.2. Block diagram of a recursive linear state estimator.

is known as the "innovations process." It contains the "fresh" new information that comes at each time k to the estimator. The Kalman filter is a linear estimator, as above, in which the gain $\mathbf{K}_k$, that multiplies the innovations process before feeding it back to form the new state estimate, is chosen in an optimal manner. That is, the gain $\mathbf{K}_k$ is selected to minimize the variance of the estimation error $\mathbf{e}_k$ as expressed by Equation 4.86. The derivation of the optimal gain $\mathbf{K}_k$ will be shown next, but before that, a few words to explain the terminology "filter." In the context of estimation theory, the word "filtering" corresponds to finding an estimation of $\mathbf{X}_k$ at time t_k, given the measurements information up to and including the time t_k. This is in contrast to

1. "Smoothing," where $\mathbf{X}_{k-j}$ for the integer $j > 0$ is estimated based on the information up to time t_k.
2. "Prediction," where $\mathbf{X}_{k+j}$ for $j > 0$ is estimated based on the information up to time t_k.

Coming back to Equation 4.86, the error covariance $\mathbf{P}_k$

$$\mathbf{P}_k = E(\mathbf{e}_k \mathbf{e}_k^T) \tag{4.88}$$

can be now computed as a function of the arbitrary $n \times l$ gain matrix $\mathbf{K}_k$ where l is the dimension of $\mathbf{Z}_k$ and n is the dimension of $\mathbf{X}_k$:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k(-)(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \tag{4.89}$$

where

$$\mathbf{P}_k(-) = E[\mathbf{e}_k(-)\mathbf{e}_k(-)^T] \tag{4.90}$$

and $\mathbf{R}_k$ is the measurement noise covariance. In the derivation of Equation 4.89 the assumption was that the error $\mathbf{e}_k(-)$ is independent of $\mathbf{V}_k$. $\mathbf{K}_k$ is chosen to minimize the following performance criterion

$$J_k = \text{tr}(\mathbf{P}_k) \tag{4.91}$$

for reasons similar to those explained in Section 4.3.2.

For such computation the following matrix identity is needed. Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices where $\mathbf{B}$ is symmetric. Then

$$\frac{\partial}{\partial \mathbf{A}}[\text{tr}(\mathbf{A}\mathbf{B}\mathbf{A}^T)] = 2\mathbf{A}\mathbf{B} \tag{4.92}$$

Thus, from Equation 4.88 and 4.91 one computes

$$\frac{\partial}{\partial \mathbf{K}_k}[\text{tr}(\mathbf{P}_k)] = 0 \tag{4.93}$$

which gives the optimal gain matrix

$$\mathbf{K}_{k,\text{opt}} = \mathbf{P}_k(-)\mathbf{H}_k^T[\mathbf{H}_k\mathbf{P}_k(-)\mathbf{H}_k^T + \mathbf{R}_k]^{-1} \tag{4.94}$$

Indeed, one can verify through the Hessian matrix of J_k that this is a minimizing solution. The only missing link in determining the complete equation of evolution of the optimal error covariance $\mathbf{P}_k$ is the relationship between $\mathbf{P}_{k-1}$ and $\mathbf{P}_k(-)$.

A simple manipulation of Equations 4.75, 4.79, and 4.82 yields

$$\mathbf{e}_k(-) = \mathbf{A}_{k-1}\mathbf{e}_{k-1} + \mathbf{w}_{k-1} \tag{4.95}$$

Using the assumption that $\mathbf{e}_k$ and $\mathbf{w}_k$ are independent, one obtains

$$\mathbf{P}_k(-) = \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1} \tag{4.96}$$

where $\mathbf{Q}_k$ is the covariance of the process noise.

Substitution of the optimal gain $\mathbf{K}_{k,\text{opt}}$ of Equation 4.94 into 4.89 yields the simple updating law:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_{k,\text{opt}}\mathbf{H}_k)\mathbf{P}_k(-) \tag{4.97}$$

Equations 4.94, 4.96 and 4.97 provide the algorithm for both updating of the estimation error covariance matrix $\mathbf{P}_k$ and determination of the optimal gain matrix $\mathbf{K}_k$ at every step.

To start the Kalman filtering algorithm, recall that the initial vector $\mathbf{X}_0$ is Gaussian with mean $\boldsymbol{\mu}_x$ and covariance $\mathbf{P}_0$. Then

$$\hat{\mathbf{X}}_0 = E(\mathbf{X}_0) = \boldsymbol{\mu}_x \tag{4.98}$$

$$\mathbf{P}_0 = E[(\mathbf{X}_0 - \hat{\mathbf{X}}_0)(\mathbf{X}_0 - \hat{\mathbf{X}}_0)^T] = E[(\mathbf{X}_0 - \boldsymbol{\mu}_x)(\mathbf{X}_0 - \boldsymbol{\mu}_x)^T] \tag{4.99}$$

Using Equation 4.96, $\mathbf{P}_1(-)$ is calculated from $\mathbf{P}_0$. Then, using Equation 4.94, $\mathbf{K}_{1,\text{opt}}$ is calculated and finally, from Equation 4.97, $\mathbf{P}_1$ is found, and so on.

It is important to note that the evolution of the error covariance matrix $\mathbf{P}_k$, representing the performance of the Kalman filter, is independent of the measured data $\mathbf{Z}_k$, $k = 0, 1, 2, \ldots$. The vector $\mathbf{P}_k$ does depend on the dynamic model parameters and on the parameters of the probabilistic model of $\mathbf{X}_0$, $\mathbf{V}_k$, and $\mathbf{w}_k$.

It is feasible, if so desired, to solve "off-line" for the matrices $\mathbf{P}_k$ and $\mathbf{K}_k$, for arbitrarily many values of k, trading a great deal of real-time computations for a much increased memory requirement.

Specializing the Kalman filtering equations to the problem of identifying a fixed Gaussian vector $\mathbf{X}$ is straightforward. The state Equation 4.75 for a constant process $\mathbf{X}$ becomes

$$\mathbf{X}_k = \mathbf{X}_{k-1} \tag{4.100}$$

$$\mathbf{X}_0 = \mathbf{X} \tag{4.101}$$

In other words, the standard identification problem is characterized by $\mathbf{A}_k = \mathbf{I}$ and by having no process noise.

The measurement model given in Equation 4.76 for the identification problem remains unchanged. Note that the measurement equation is time varying as the matrix $\mathbf{H}_k$ needs not be the same from one value of k to another.

Many of the filtering equations now become very simple. For instance

$$\hat{\mathbf{X}}_k(-) = \hat{\mathbf{X}}_{k-1} \tag{4.102}$$

$$\mathbf{P}_k(-) = \mathbf{P}_{k-1} \tag{4.103}$$

The resulting recursive identification equations of a Gaussian vector $\mathbf{X}$ having mean $\boldsymbol{\mu}_x$ and covariance $\mathbf{P}_0$, are as follows:

$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_{k-1} + \mathbf{P}_{k-1}\mathbf{H}_k^T[\mathbf{H}_k\mathbf{P}_{k-1}\mathbf{H}_k^T + \mathbf{R}_k]^{-1}[\mathbf{Z}_k - \mathbf{H}_k\hat{\mathbf{X}}_{k-1}] \tag{4.104}$$

$$\hat{\mathbf{X}}_0 = \boldsymbol{\mu}_x \tag{4.105}$$

where $\boldsymbol{\mu}_x$ is typically zero.

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \mathbf{P}_{k-1}\mathbf{H}_k^T[\mathbf{H}_k\mathbf{P}_{k-1}\mathbf{H}_k^T + \mathbf{R}_k]^{-1}\mathbf{H}_k\mathbf{P}_{k-1} \tag{4.106}$$

Equation 4.106 shows the evolution of the estimation error covariance. Since it is independent of any real-time data, it can be solved or analyzed a priori to provide the full characterization of the filter performance features such as the steady-state error covariance and the speed of convergence to steady state conditions.

Characterizing an equilibrium solution to Equation 4.106 as follows:

$$\lim_{k\to\infty} \mathbf{P}_{k-1} = \lim_{k\to\infty} \mathbf{P}_k = \mathbf{P}_\infty \tag{4.107}$$

then an obvious steady-state solution $\mathbf{P}_\infty$ of Equation 4.106 is

$$\mathbf{P}_\infty = 0 \tag{4.108}$$

indicating that as the sequence of measurements $\mathbf{Z}_k$ grows longer, the estimation accuracy gets better and better, if only the matrices $\mathbf{H}_k$ are such that $\mathbf{P}_k$ converges to zero. The second term on the right-hand side of Equation 4.106 is the one that determines the updating of $\mathbf{P}_k$. Intuitively, the larger the intensity of the measurement noise, (i.e., the elements of the matrix $\mathbf{R}_k$) the slower is the convergence of $\mathbf{P}_k$ to a steady state solution.

The property of zero steady-state error covariance is entirely attributed to the lack of process noise in Equation 4.100. Equations 4.94, 4.96, and 4.97 in general produce under certain controllability and observability assumptions of the model, a nonzero steady-state error covariance $\mathbf{P}_\infty$.

For example, let the process equation be

$$\mathbf{X}_k = \mathbf{X}_{k-1} + \mathbf{w}_{k-1}$$
$$\mathbf{X}_0 = \mathbf{X} \tag{4.109}$$

Here a vector $\mathbf{X}_k$ is nominally fixed but its value is subject to random fluctuations modeled by a white noise sequence $\mathbf{w}_k$ (zero mean and covariance $\mathbf{Q}_k$). It can be shown that for this case the filtering equations are

$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_{k-1} + \mathbf{K}_k(\mathbf{Z}_k - \mathbf{H}_k\hat{\mathbf{X}}_{k-1}) \tag{4.110}$$

$$\mathbf{K}_k = (\mathbf{P}_{k-1} + \mathbf{Q}_{k-1})\mathbf{H}_k^T[\mathbf{H}_k(\mathbf{P}_{k-1} + \mathbf{Q}_{k-1})\mathbf{H}_k^T + \mathbf{R}_k]^{-1} \tag{4.111}$$

$$\mathbf{P}_k = \mathbf{P}_{k-1} + \mathbf{Q}_{k-1} - \mathbf{K}_k\mathbf{H}_k(\mathbf{P}_{k-1} + \mathbf{Q}_{k-1}) \tag{4.112}$$

Considering for a moment a time-invariant case $\mathbf{Q}_k = \mathbf{Q}$, $\mathbf{R}_k = \mathbf{R}$, and $\mathbf{H}_k = \mathbf{H}$, then a steady state solution $\mathbf{P}_\infty$ of Equation 4.112 satisfies the algebraic equation:

$$(\mathbf{P}_\infty + \mathbf{Q})\mathbf{H}^T[\mathbf{H}(\mathbf{P}_\infty + \mathbf{Q})\mathbf{H}^T + \mathbf{R}]^{-1}\mathbf{H}(\mathbf{P}_\infty + \mathbf{Q}) = \mathbf{Q} \tag{4.113}$$

In the time-varying case, a constant steady-state solution $\mathbf{P}_\infty$ may not exist at all.

The initial error covariance matrix $\mathbf{P}_0$ reflects the a priori knowledge regarding the expected axis misalignment at every robot joint. One may take worst case values based on models of the tolerances in the robot axes of motion. If such models are not easily available, one may take an arbitrary positive definite and symmetric initial matrix $\mathbf{P}_0$. Under proper operation of the filter and after sufficiently many measurements, the value of $\mathbf{P}_k$, the error covariance, converges to a steady-state matrix $\mathbf{P}_\infty$. In the case of fixed unknown state $\mathbf{X}$, as was just shown, $\mathbf{P}_\infty = 0$. Thus, after sufficiently many measurements the filter's estimate is stabilized as less and less weighting is given to new measurements. Of course, at that stage there is no sense in taking more measurements. The speed of convergence to such steady state may be found by off-line analysis of the data-independent Riccati Equation 4.106. In particular, the sufficient number of measurements depends on the initial uncertainity $\mathbf{P}_0$, the particular robot configurations $\mathbf{H}_k$ and the measurement noise covariances $\mathbf{R}_k$.

In the case of arbitrary assignment of $\mathbf{P}_0$ a reasonable assumption is that axis misalignments in different robot joints are independent of each other. The errors in the kinematic parameters within each link, however, are generally highly dependent. Thus, rather than taking $\mathbf{P}_0$ to be a diagonal matrix, a block diagonal structure better reflects reality here.

Mathematically, the maximum number of measurements, M, is determined from the inequality

$$\|\mathbf{P}(j) - \mathbf{P}_\infty\| \leq \delta, \quad \forall j \geq M \tag{4.114}$$

where the number $\delta > 0$ denotes a prespecified desired proximity to filter steady state.

As was previously shown, $\mathbf{P}_\infty = 0$ due to the lack of "process noise." In other words under the assumption that the only imperfection of the robot joints is the constant vector $\mathbf{X}$ of kinematic parameter changes, the calibration can be done infinitely accurately regardless of the measurement noise if only a large enough set of measurements is taken.

Practically though, the robot accuracy cannot become better than the robot repeatability. Possible sources of nonzero repeatability are joint clearance effects, gear backlash, joint position transducer noise, and imperfect feedback and feedforward control actions. These may be modeled via introduction of process noise terms to the $\mathbf{X}_k$ equation as was shown in Equation 4.109. This results in convergence to a nonzero steady-state error covariance $\mathbf{P}_\infty$ or, in other words, there will exist a nonzero lower bound on the calibration error. $\mathbf{P}_\infty$ is a solution of the algebraic Riccati Equation 4.113. $\mathbf{P}_\infty$ depends on both the process and measurement noise and may provide a useful clue regarding the interdependence of calibration accuracy, measurement noise, and robot repeatability. It should be noted that due to the time-varying nature of the system, a constant matrix $\mathbf{P}_\infty$ may not exist, but one may find lower and upper bounds on the norm of $\mathbf{P}_k$ as $k \to \infty$.

As measurements $\mathbf{Z}(j)$ are taken at arbitrary robot configurations and what appears to be an arbitrary ordering, interesting questions may be raised, such as

1. Are there robot measurement configurations $\mathbf{H}(j)$ that do not contribute useful information for the identification process?
2. Are there preferred measurement configurations $\mathbf{H}(j)$ that will provide the fastest convergence of the identification algorithm?
3. How much "excitation" of the robot joints is needed for proper identification? In other words, when the robot is moved from one measurement configuration to another, do we need to move all the joints, or will it do to move one joint at a time?
4. What is the minimum numer M of measurements for a given predefined sequence of specific configurations $\mathbf{H}(1)$, $\mathbf{H}(2)$, ... ? Can lower or upper bounds on such a number be derived when a desired final value $\mathbf{P}(n)$ of the error covariance matrix $\mathbf{P}$ is given?

Finally, an important comment should be made regarding the practical operation of Kalman filters. The entire previous analysis was based on an assumption of no modeling errors. In practice, the matrices $\mathbf{A}_k$, $\mathbf{H}_k$, $\mathbf{Q}_k$, $\mathbf{R}_k$, and $\mathbf{P}_0$ and the vector $\boldsymbol{\mu}_x$ may not be known precisely. Such modeling errors may even cause filter instability. The following property of Kalman filters, that may be easily verified, is helpful. When no modeling errors are present, the innovations process $\mathbf{v}_k$ (Equation 4.87) is a white noise sequence (i.e., $\mathbf{v}_k$ is zero mean Gaussian and the samples $\{\ldots, \mathbf{v}_{k-1}, \mathbf{v}_k, \mathbf{v}_{k+1}, \ldots\}$ are uncorrelated). By statistically testing the process $\mathbf{v}_k$ for zero mean and uncorrelated samples one may reveal the presence of modeling errors. One can then better tune the filter internal model to match the actual model of the physical system.

4.4 NONLINEAR LEAST-SQUARES ESTIMATION

The identification of the kinematic parameter errors using linear least-squares techniques requires obtaining a linearized error model where the unknown errors are related to the measurement error through the identification Jacobian. A "small perturbation" requirement is crucial to guarantee convergence of the least-squares algorithm whenever it is done iteratively. The following questions then arise:

1. Can the identification be accomplished without the use of the identification Jacobian, using only the nonlinear kinematic model?
2. Viewing the search for the kinematic parameter errors as a standard unconstrained nonlinear optimization problem, can the identification Jacobian, as well as higher order derivatives of the nonlinear kinematic functions, be used in a manner that improves on the speed of convergence and the quality of the identification?
3. What can be done to improve on the convergence properties of the identification algorithm in cases where the kinematic parameter errors are not necessarily small?

This section attempts to provide answers to these questions. Section 4.4.1 is a discussion of direct search methods that indeed have minimal modeling requirements, but may be much slower as compared to more sophisticated methods.

The final section, Section 4.4.2, focuses on gradient methods, where again the identification Jacobian is used, but in a manner that differs from that of the linear least-squares algorithm. Some of the more popular methods such as the Levenberg-Marquardt algorithm are introduced. The section touches on the basic ideas behind small as well as large residual optimization methods as applied to the calibration problem.

4.4.1 Direct Search Methods

The vector form of the robot kinematic model (Equation 4.8) may be written as

$$\mathbf{x}_j = \mathbf{f}(\mathbf{q}_j, \mathbf{k}) \qquad j = 1, 2, \ldots, m \tag{4.115}$$

where $\mathbf{x}_j$ is the measured pose vector at the measurement configuration j and $\mathbf{q}_j$ are the corresponding joint variables.

$$\mathbf{k} = \mathbf{k}_0 + \delta\mathbf{k} \tag{4.116}$$

$$\mathbf{q}_j = \mathbf{q}_{j0} + \delta\mathbf{q} \tag{4.117}$$

where $\mathbf{k}_0$ is the nominal set of kinematic parameters and $\delta\mathbf{k}$ is the unknown vector of kinematic parameter errors, not including the joint offsets. The vector $\mathbf{q}_{j0}$ contains the nominal joint commands where $\delta\mathbf{q}$ is the vector of joint offsets.

The number of measurement configurations, m, needs to be large enough so that

$$\dim \mathbf{x}_{ag} = \dim(\mathbf{x}_1^T, \ldots, \mathbf{x}_m^T) = 6m > \dim(\mathbf{q}) + \dim(\mathbf{k}) = \mathbf{K} \tag{4.118}$$

where the $6m$ vector $\mathbf{x}_{ag}$ is the aggregation of the measured pose vectors.

$$\mathbf{x}_{ag} = \mathbf{f}_{ag}(\mathbf{q}_1, \ldots, \mathbf{q}_m, \mathbf{k}) = \begin{bmatrix} \mathbf{f}(\mathbf{q}_1, \mathbf{k}) \\ \vdots \\ \mathbf{f}(\mathbf{q}_m, \mathbf{k}) \end{bmatrix} \tag{4.119}$$

For an N degree of freedom manipulator and a DH modeling convention, we have that $K = 4N$.

The optimization problem is to select $\delta\mathbf{k}$ and $\delta\mathbf{q}$ to minimize a cost function L defined as

$$\begin{aligned} L &= \|\mathbf{x}_{ag} - \mathbf{f}_{ag}(\mathbf{q}_1 + \delta\mathbf{q}, \ldots, \mathbf{q}_m + \delta\mathbf{q}, \mathbf{k} + \delta\mathbf{k})\|_2 \\ &= [\mathbf{x}_{ag} - \mathbf{f}_{ag}(\mathbf{q}_1 + \delta\mathbf{q}, \ldots, \mathbf{q}_m + \delta\mathbf{q}, \mathbf{k} + \delta\mathbf{k}]^T [\mathbf{x}_{ag} - \mathbf{f}_{ag}(\mathbf{q}_1 + \delta\mathbf{q}, \ldots, \mathbf{q}_m \\ &\quad + \delta\mathbf{q}, \mathbf{k} + \delta\mathbf{k})] \\ &= \mathbf{e}_{ag}^T(\delta\mathbf{k}_{ag})\mathbf{e}_{ag}(\delta\mathbf{k}_{ag}) \end{aligned} \tag{4.120}$$

where $\mathbf{e}_{ag}$ is an aggregated pose error vector and $\delta\mathbf{k}_{ag} = (\delta\mathbf{q}^T, \delta\mathbf{k}^T)^T$ combines together all the kinematic error parameters.

If no derivatives of $\mathbf{f}(\cdot)$ with respect to $\mathbf{q}$ or $\mathbf{k}$ are available, the optimization of L may be accomplished through direct search varying the components of $\delta\mathbf{k}_{ag}$ one at a time and solving the forward kinematics repeatedly.

A search algorithm based on the "Golden Section" method [18] is carried out as follows:

Step 1: Define $\delta\mathbf{k}_{ag} = (\delta_1, \ldots, \delta_K)^T$ and let the initial value $\delta\mathbf{k}_{ag}^{(0)}$ be zero.

Step 2: Conduct a line search varying the first component of $\delta\mathbf{k}_{ag}$. Let

$$\delta\mathbf{k}_{ag}^{(1)} = (\delta_1, 0, \ldots, 0)^T \tag{4.121}$$

Decide on a search interval for the scalar variable δ_1 as given by

$$\delta_{1\,\min} \leq \delta_1 \leq \delta_{1\,\max} \tag{4.122}$$

The interval $[\delta_{1\,\min}, \delta_{1\,\max}]$ is then scanned using the golden section method as follows:

- The initial width of uncertainty, d_1, is

$$d_1 = \delta_{1\,\max} - \delta_{1\,\min} \tag{4.123}$$

- Let d_2 be the width of uncertainty at the second step.

$$d_2 = \frac{2}{1+\sqrt{5}} d_1 \tag{4.124}$$

- Evalute L at

$$\delta_{1A} = \delta_{1\,\max} - d_2 \tag{4.125}$$

and at

$$\delta_{1B} = \delta_{1\,\min} + d_2 \tag{4.126}$$

- If $L(\delta_{1A}) < L(\delta_{1B})$, then form an interval of uncertainty $[\delta_{1\,\min}, \delta_{1B}]$. If $L(\delta_{1A}) > L(\delta_{1B})$, then form an interval of uncertainty $[\delta_{1A}, \delta_{1\,\max}]$. In other words, the new interval of uncertainty of width d_2 contains the point that has the smallest value of L.
- Let

$$d_3 = \frac{2}{1+\sqrt{5}} d_2 \tag{4.127}$$

- Evaluate L at the symmetric points that are spaced d_3 from the boundaries of the interval of uncertainty as in the previous steps.

The process repeats itself with smaller and smaller intervals of uncertainty. The iterations may be stopped when a convenient termination condition has been reached. We now denote δ_1^* as the optimal δ_1 value, obtained in the search described above.

Step 3: Now vary the second variable, δ_2, in the same manner while keeping the previously obtained δ_1^* value.

$$\delta \mathbf{k}_{ag}^{(2)} = (\delta_1^*, \delta_2, \ldots, 0) \tag{4.128}$$

and repeat the golden section search with respect to δ_2.

Step 4: Proceed, varying the variables one by one, in the following order

$$\delta \mathbf{k}_{ag}^{(j)} = (\delta_1^*, \delta_2^*, \ldots, \delta_{j-1}^*, \delta_j, 0, \ldots, 0) \tag{4.129}$$

Finally, after optimizing the last variable, denoted here as δ_K, we will have

$$\delta \mathbf{k}_{ag}^{(K)} = (\delta_1^*, \ldots, \delta_K^*) \tag{4.130}$$

Repeat the whole process cyclically optimizing again for δ_1

$$(\delta_1^{**}, \delta_2^*, \ldots, \delta_{4N}^*) \rightarrow (\delta_1^{**}, \delta_2^{**}, \ldots, \delta_{4N}^*) \rightarrow \cdots \tag{4.131}$$

and so on.

Various search routines differ from one another in the manner in which the sequence of line search problems are ordered. Convergence of all direct search algorithms is naturally very slow. With a direct search, however, no Jacobians need to be calculated.

4.4.2 Gradient Methods

In most practical calibration problems, the constraints on the kinematic parameter errors can be ignored. Thus the parameter estimation may be treated as an unconstrained nonlinear optimization problem. Let $\delta\mathbf{k}_{ag} = (\delta_1, \ldots, \delta_K)^T$. The problem is the minimization of $L(\delta_1, \ldots, \delta_K)$ as defined in Equation 4.120. The quadratic structure of L vastly simplifies the solutions as will be seen later in this section.

Most nonlinear least-square methods are iterative. Let $(\delta\mathbf{k}_{ag})_j$ denote the estimate of the optimal solution at the beginning of the jth iteration. The jth iteration consists, in general, of the computation of a search vector $\mathbf{p}_j$ from which the new estimate $(\delta\mathbf{k}_{ag})_{j+1}$ is obtained according to

$$(\delta\mathbf{k}_{ag})_{j+1} = (\delta\mathbf{k}_{ag})_j + \alpha_j\mathbf{p}_j \tag{4.132}$$

where α_j is a scalar obtained by means that vary from one optimization method to another. The methods of selecting the search direction vary in different optimization techniques.

A basic requirement of any iterative search is that the performance measure improves from one iteration to the next:

$$L[(\delta\mathbf{k}_{ag})_{j+1}] < L[(\delta\mathbf{k}_{ag})_j] \tag{4.133}$$

which requires when expanding $L[(\delta\mathbf{k}_{ag})_{j+1}]$ into a Taylor series, that

$$\mathbf{g}_j^T\mathbf{p}_j < 0 \tag{4.134}$$

where $\mathbf{g}_j$ is the gradient of L evaluated at $\delta\mathbf{k}_{ag} = (\delta\mathbf{k}_{ag})_j$. Every vector $\mathbf{p}_j$ that satisfies Equation 4.134 provides what is called a descent direction. Obviously, for small values of α_j, the greatest reduction in the performance function value is obtained in the direction opposite to that of the gradient $\mathbf{g}_j$.

$$\mathbf{p}_j^* = -\mathbf{g}_j \tag{4.135}$$

Such a search vector is said to have a steepest descent direction.

For an arbitrary choice of search vector $\mathbf{p}_j$, not necessarily along the direction of steepest descent, and a given value of $(\delta\mathbf{k}_{ag})_j$, the coefficient α_j may be optimized to minimize $L[(\delta\mathbf{k}_{ag})_j + \alpha_j\mathbf{p}_j]$. The result of such an optimization is that at $\alpha_j = \alpha_j^*$.

$$\mathbf{g}_{j+1}^T\mathbf{p}_j = 0 \tag{4.136}$$

The vector $\mathbf{g}_{j+1}$ is defined as

$$\mathbf{g}_{j+1} = \nabla L[(\delta \mathbf{k}_{ag})_j + \alpha_j^* \mathbf{p}_j] \tag{4.137}$$

In other words, theoretically one should move along the search vector to the point where the new gradient vector becomes orthogonal to the search vector. This strategy of choosing α_j is computationally intensive and, therefore, not often implemented in practice.

At this point it is important to review the necessary and sufficient conditions for a minimum.

If a point $(\delta \mathbf{k}_{ag})^*$ is a minimum, then

$$\mathbf{g}[(\delta \mathbf{k}_{ag})^*] = 0 \tag{4.138}$$

At such a point, a sufficient condition for minimum is that the Hessian matrix, $\mathbf{G}$, is positive definite.

$$\mathbf{G}[(\delta \mathbf{k}_{ag})^*] = \nabla^2 L(\delta \mathbf{k}_{ag})|_{(\delta \mathbf{k}_{ag})^*} > 0 \tag{4.139}$$

The first two terms in the Taylor series expansion of $\mathbf{g}[(\delta \mathbf{k}_{ag})_j + \mathbf{p}_j]$ are

$$\mathbf{g}[(\delta \mathbf{k}_{ag})_{j+1}] = \mathbf{g}[(\delta \mathbf{k}_{ag})_j + \mathbf{p}_j] \simeq \mathbf{g}[(\delta \mathbf{k}_{ag})_j] + \mathbf{G}[(\delta \mathbf{k}_{ag})_j]\mathbf{p}_j \tag{4.140}$$

The right-hand side of Equation 4.140 is exact if $L[(\delta \mathbf{k}_{ag})_j]$ is quadratic in the components of $(\delta \mathbf{k}_{ag})_j = (\delta_1, \ldots, \delta_K)$. In such a case, by choosing

$$\mathbf{p}_j = \mathbf{G}^{-1}[(\delta \mathbf{k}_{ag})_j]\mathbf{g}[(\delta \mathbf{k}_{ag})_j] \tag{4.141}$$

an optimum $\mathbf{g}^*[(\delta \mathbf{k}_{ag})_{j+1}]$ is obtained in only one iteration. If $L(\cdot)$ is nonquadratic in $(\delta_1, \ldots, \delta_K)$, more iterations are needed. The updating strategy as presented in Equations 4.140 and 4.141 constitute Newton's method. If the Hessian matrix, $\mathbf{G}(\cdot)$, is known, the most practical way of evaluating $\mathbf{p}_j$ (rather than matrix inversion as in Equation 4.141) is to solve the following set of linear algebraic equations:

$$\mathbf{G}[(\delta \mathbf{k}_{ag})_j]\mathbf{p}_j = -\mathbf{g}[(\delta \mathbf{k}_{ag})_j] \tag{4.142}$$

using available routines.

Returning now to the particular performance measure L chosen for the robot calibration problem (Equation 4.120), its particular sum of squares structure provides useful relationships between the Hessian matrix and the identification Jacobian matrix.

Considering the robot kinematics (Equations 4.115), let $\mathbf{J}_j$ denote the identification Jacobian at configuration $\mathbf{q}_j$. That is

$$\mathbf{J}_j = \left[\frac{\partial \mathbf{f}(\mathbf{q},\mathbf{k})}{\partial \mathbf{q}_1},\ldots,\frac{\partial \mathbf{f}(\mathbf{q},\mathbf{k})}{\partial \mathbf{q}_N},\frac{\partial \mathbf{f}(\mathbf{q},\mathbf{k})}{\partial \mathbf{k}_1},\ldots,\frac{\partial \mathbf{f}(\mathbf{q},\mathbf{k})}{\partial \mathbf{k}_{K-N}}\right]\Bigg|_{q=q_j+\delta q,\,\mathbf{k}=\mathbf{k}_0+\delta \mathbf{k}}$$
$$= \left[\frac{\partial \mathbf{f}(\mathbf{q},\mathbf{k})}{\partial \delta_1},\ldots,\frac{\partial \mathbf{f}(\mathbf{q},\mathbf{k})}{\partial \delta_K}\right]\Bigg|_{q=q_j+\delta q,\,\mathbf{k}=\mathbf{k}_0+\delta \mathbf{k}} \tag{4.143}$$

Aggregating all measurement configurations, $j = 1,\ldots,m$, let $\mathbf{J}$ be the aggregated identification Jacobian corresponding to measurement configurations $\mathbf{q}_1,\ldots,\mathbf{q}_m$.

$$\mathbf{J} = \begin{bmatrix}\mathbf{J}_1\\ \vdots \\ \mathbf{J}_m\end{bmatrix} \tag{4.144}$$

To obtain an expression for the gradient vector of

$$L(\delta \mathbf{k}_{ag}) = \mathbf{e}_{ag}^T(\delta \mathbf{k}_{ag})\mathbf{e}_{ag}(\delta \mathbf{k}_{ag}) \tag{4.145}$$

note that the first partial derivative with respect to $\delta_j, j = 1,\ldots,K$, is given by

$$\mathbf{g}_j = \frac{\partial L(\delta \mathbf{k}_{ag})}{\partial \delta_j} = 2\sum_{i=1}^{6m}[\mathbf{x}_{ag} - \mathbf{f}_{ag}(\delta \mathbf{k}_{ag})]_i \frac{\partial[\mathbf{x}_{ag} - \mathbf{f}_{ag}(\delta \mathbf{k}_{ag})]_i}{\partial \delta_j} \tag{4.146}$$

Using the definition of $\mathbf{J}$ in Equations 4.143 and 4.144, the gradient vector can be written as

$$\mathbf{g}(\delta \mathbf{k}_{ag}) = -2\mathbf{J}^T\mathbf{e}_{ag} \tag{4.147}$$

where both $\mathbf{J}$ and $\mathbf{e}_{ag}$ depend on $\delta \mathbf{k}_{ag}$.

Differentiating Equation 4.146 with respect to δ_k gives the kj element of the Hessian matrix

$$(\mathbf{G})_{kj} = 2\sum_{i=1}^{6m}\left[\frac{\partial(\mathbf{e}_{ag})_i}{\partial \delta_k}\frac{\partial(\mathbf{e}_{ag})_i}{\partial \delta_j} + (\mathbf{e}_{ag})_i\frac{\partial^2(\mathbf{e}_{ag})_i}{\partial \delta_k \partial \delta_j}\right] \tag{4.148}$$

Denoting by $\mathbf{T}_i$ the Hessian matrix of $(\mathbf{e}_{ag})_i$, the ith column of $\mathbf{e}_{ag}$

$$\mathbf{T}_i(\delta \mathbf{k}_{ag}) = \nabla^2[(\mathbf{e}_{ag})_i] \tag{4.149}$$

then the complete Hessian matrix of $L(\delta \mathbf{k}_{ag})$ can be written as

$$\mathbf{G} = 2\mathbf{J}^T\mathbf{J} + 2\mathbf{S} \tag{4.150}$$

where $\mathbf{G}$, $\mathbf{J}$, and $\mathbf{S}$ all depend on $\delta \mathbf{k}_{ag}$. The matrix $\mathbf{S}$ is defined as

$$\mathbf{S} = \sum_{i=1}^{6m} (\mathbf{e}_{ag})_i \mathbf{T}_i \tag{4.151}$$

Equations 4.150 and 4.151 display the structure of the derivatives of the sum of squares performance function. Substitution of the gradient expression in Equation 4.147 and the Hessian matrix given in Equation 4.150 into Newton's algorithm at the jth iteration for finding $\delta\mathbf{k}_{ag}$ results in the following equation for the search vector $\mathbf{p}_j$

$$(\mathbf{J}^{(j)^T}\mathbf{J}^{(j)} + \mathbf{S}^{(j)})\mathbf{p}_j = \mathbf{J}^{(j)^T}\mathbf{e}_{ag}^{(j)} \tag{4.152}$$

where $\mathbf{e}_{ag}^{(j)}$ is the aggregated pose vector at the jth identification iteration and $\mathbf{J}^{(j)}$ and $\mathbf{S}^{(j)}$ depend on $\delta\mathbf{k}_{ag}$ of the jth iteration. The updated estimate is

$$(\delta\mathbf{k}_{ag})_{j+1} = (\delta\mathbf{k}_{ag})_j + \mathbf{p}_j \tag{4.153}$$

The computation of $\mathbf{S}[(\delta\mathbf{k}_{ag})_j]$ involves the specification and evaluation of $3mK(K+1)$ second derivative terms. The remainder of the Hessian matrix, however, is expressed in terms of first derivatives only. This observation gives rise to two broad classes of special algorithms for nonlinear least-squares parameter estimation. Those that ignore the term $\mathbf{S}[(\delta\mathbf{K}_{ag})_j]$ are useful for small residual cases. These are the cases where $\|\delta\mathbf{k}_{ag}\|$ is small. The other class, applicable to larger residual cases, involves the approximation of $\mathbf{S}[(\mathbf{k}_{ag})_j]$ in some way.

By neglecting the term $\mathbf{S}[(\mathbf{k}_{ag})_j]$ in the Newton equation, (Equation 4.152), the identification algorithm reduces to the familiar iterative linear least-squares algorithm, also referred to as the *Gauss–Newton method*

$$\mathbf{J}^T[(\delta\mathbf{k}_{ag})_j]\mathbf{J}[(\delta\mathbf{k}_{ag})_j]\mathbf{p}_j = \mathbf{J}^T[(\delta\mathbf{k}_{ag})_j]\mathbf{e}_{ag}^{(j)} \tag{4.154}$$

combined with the estimate updating equation, Equation 4.153.

Even if $\mathbf{J}^T[(\delta\mathbf{k}_{ag})_j]\mathbf{J}[(\delta\mathbf{k}_{ag})_j]$ is nonsingular, Equation 4.154 does not necessarily guarantee that $\mathbf{p}_j$ is a descent direction since the scalar coefficient α_j, which is assumed to be 1 in Equation 4.153, might be too large a step along the search vector.

The Gauss–Newton algorithm breaks down at the singularities of the identification Jacobian and converges very slowly near such singularities. On the other hand, the Gauss–Newton method tends to Newton's method, the closer the estimate is to the correct set of kinematic parameter errors. In other words, if $\mathbf{e}_{ag}(\delta\mathbf{k}_{ag}) \to 0$ as $\delta\mathbf{k}_{ag} \to \delta\mathbf{k}_{ag}^*$, then $\mathbf{S}(\delta\mathbf{k}_{ag})$ gets smaller and smaller. As is well known [24], Newton's method exhibits a second-order rate of convergence. That is, $M = 2$ is the largest value of M for which the following convergence ratio exists

$$R = \lim_{j\to\infty} \frac{\|(\delta\mathbf{k}_{ag})_{j+1} - \delta\mathbf{k}_{ag}^*\|}{\|(\delta\mathbf{k}_{ag})_j - \delta\mathbf{k}_{ag}^*\|^M} \tag{4.155}$$

Rapid convergence rates are associated with large values of M and small values of R.

The *Levenberg–Marquardt* (LM) method was introduced as an improvement to the Gauss–Newton method. The technique is designed to overcome problems related to singularity of the matrix $\mathbf{J}^T\mathbf{J}$, while maintaining the Gauss–Newton convergence properties near the optimal solution. The idea is to modify Equation 4.154 by adding a time varying nonnegative scalar coefficient, μ_j, as follows

$$[\mathbf{J}^T[(\delta\mathbf{k}_{ag})_j]\mathbf{J}[(\delta\mathbf{k}_{ag})_j] + \mu_j\mathbf{I}]\mathbf{p}_j = \mathbf{J}^T[(\delta\mathbf{k}_{ag})_j]\mathbf{e}_{ag}^{(j)} \tag{4.156}$$

where $\mathbf{I}$ is the $K \times K$ identity matrix. For a sufficiently large value of μ_j, the matrix $\mathbf{J}^T\mathbf{J} + \mu_j\mathbf{I}$ is positive definite, and $\mathbf{p}_j$ can be made a descent direction. As the solution approaches the optimum, $\delta\mathbf{k}_{ag} \to \delta k_{ag}^*$, the scalar μ_j is adjusted to have smaller and smaller values so that the method acquires the asymptotic rate of convergence of the Gauss–Newton method.

There are many possible strategies for selecting μ_j at each iteration. Two methods are described next. One should observe that as $\mu_j \to \infty$, the effect of the $\mu_j\mathbf{I}$ term increasingly dominates that of $\mathbf{J}^T\mathbf{J}$ so that $\mathbf{p}_j \to \mu_j^{-1}\mathbf{J}^T[(\delta\mathbf{k}_{ag})_j]\mathbf{e}_{ag}$, which represents an infinitesimal step in the steepest descent direction.

Method 1 (Levenberg [17]): The variable μ_j is chosen through linear search to minimize $L[(\delta\mathbf{k}_{ag})_j + \mathbf{p}_j]$, for $\mathbf{p}_j$ given by Equation 4.156. Everything else is held constant. Thus, by choosing a sufficiently large value for μ_j, the optimization method can be globally convergent.

Note that for every new value of μ_j, the system of linear equations for $\mathbf{p}_j$ (Equation 4.156), has to be resolved. This makes the method somewhat impractical.

Method 2 (Marquardt [19]): The variable μ_j is initially set to some positive value (say, $\mu_j = 0.01$), and a factor $v > 1$ (say, $v = 10$) is established. The factor, v, is used to increase or decrease μ_j. At the beginning of each iteration, μ_j is reduced by the factor v in an attempt to push the algorithm closer to the Gauss–Newton method. At each iteration, the cost function is evaluated and compared against the cost function value at the previous iteration. If, for the chosen value of μ_j, no reduction in the cost function is achieved, the value of μ_j is increased by the factor v. If this still fails to give a reduction in the performance function value, the new μ_j is repeatedly increased by the factor v until a reduction is obtained. More systematic methods for selection of μ_j and v are described in the text by Scales [24].

In the case where no derivative information is available, one may still apply the Gauss–Newton or LM strategies by replacing the identification Jacobian matrix, $\mathbf{J}$, with a finite difference approximation. Alternately, one may search using Newton's method for a matrix that approximates the identification Jacobian. For more details on this so called *quasi-Newton* method, the reader is again referred to Scales [24].

The LM algorithm may work surprisingly well even for large residual problems although in such cases the rate of convergence may be quite slow. Improvements through finite difference or quasi-Newton approximations of the matrix $S(\delta \mathbf{k}_{ag})_j$ of Equation 4.152 have been suggested in the literature. The details of these approaches are outside the scope of this text. Interested readers are referred to Scales [24].

4.5 OBSERVATION STRATEGY FOR ROBOT KINEMATIC IDENTIFICATION

Observation strategy for calibration refers to the selection of robot configurations and the number of observations to be made during the calibration experiment. The selection of measurement configurations during robot calibration plays an important role in determining the accuracy and speed of convergence of the least-squares identification algorithms. The kinematic parameter errors are not all equally observable. The "visibility" of each unknown parameter varies from one robot configuration to another. There may even be configurations in which some of the kinematic parameters are not observable at all.

Paying close attention to the observability issue and planning the measurement strategy accordingly may save on the total number of different robot configurations that are needed to be attained to obtain an accurate identification. It is further argued [4, 21] that there exist robot configurations at which the impact of geometric errors on the total accuracy overshadows the accuracy error attributed to nongeometric errors. Thus, by proper selection of measurement configurations the effects of unmodeled errors on the identification of the geometric parameters may become less significant.

The synthesis of an "optimal" observation strategy for a given manipulator in the presence of noise and uncertainty is as yet an unanswered research issue. The procedures described in this section could help in deciding the observation strategy for given robots while at the same time providing a measure of confidence in the results of the parameter identification.

For a given robot and measurement method we start by estimating the accuracy of the measurement system, the measurement noise, the resolution and uncertainty for the robot encoders, and the approximate range of motion of the robot joints during the observations. We then do a series of simulations tabulating the effects of strategy, number of observations, and joint range of motion on the accuracy of identification. We also note measures such as the condition number or the observability index of the identification Jacobian for each case.

The range of measures observed will depend on the details of the particular calibration problem. The units used for lengths and angles, the dimensions of the manipulator, the type of measurement method, and the scaling of the rows and columns of the Jacobian will all affect the range of measures observed. The simulations done before the experiment will help to know what range of measures apply to the problem being addressed.

The simulations should show if the proposed experimental setup does have the potential to identify the robot parameters to the accuracy desired. If necessary, the experimental method may be modified at this stage itself based on the results of the simulation. For example, the offset of the tool from the wrist of a robot may need to be increased so as to improve the observability of the wrist parameters. We then conduct the actual calibration experiment using the best observation strategy permitted by the constraints of the measurement, and a number of observations decided from the results of the simulation tables. We compute the measures for the experiment using the actual robot configurations of the observations and the nominal kinematics. This experimental condition number may now be used to refer back to the simulation tables and estimate the quality of the observation strategy during the experiment, and suggest a level of confidence in the parameters identified.

Two groups of references provide the literature background for this section. The first approach [7] focuses on the familiar numerical analysis concept of "condition number" of a matrix. Such a number provides invaluable information regarding the sensitivity of the least-squares identification algorithm to modeling errors and noise in the measured data. The key ideas are described in detail in Sections 4.5.1 and 4.5.2.

The second approach [21] adopts an observability index as a performance measure. There are similarities to the condition number of the previous section. The method is described in Section 4.5.3. Section 4.5.4 provides more insight into the question of useful and informative measurement configurations using the Kalman filtering formulation.

4.5.1 Numerical Sensitivity of Least-Squares Identification Algorithms

The solution of a linear least-squares identification problem in general amounts to the solution of an algebraic equation

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.157}$$

where $\mathbf{A}$ is a nonsingular matrix of order n. The vector $\mathbf{b}$ corresponds to the measured data and as such is subject to uncertainty. The matrix $\mathbf{A}$ is related to the measurement configuration and the system model and may itself be subject to errors. Thus:

$$\mathbf{x} + \delta\mathbf{x} = (\mathbf{A} + \delta\mathbf{A})^{-1}(\mathbf{b} + \delta\mathbf{b}) \tag{4.158}$$

where the solution error $\delta\mathbf{x}$ reflects the sensitivity of the identification algorithm to errors in the model and the data characterized in Equation 4.158 as $\delta\mathbf{A}$ and $\delta\mathbf{b}$.

As is well known to numerical analysts [11] and as shown next, such sensitivity may be characterized in terms of a single number derived from the matrix $\mathbf{A}$.

We start with a brief review of the "norm" concept. The norm of a vector $\mathbf{x}$, denoted as $\|\mathbf{x}\|$ is a function that assigns a real number to any vector $\mathbf{x}$ and

The LM algorithm may work surprisingly well even for large residual problems although in such cases the rate of convergence may be quite slow. Improvements through finite difference or quasi-Newton approximations of the matrix $S(\delta \mathbf{k}_{ag})_j$ of Equation 4.152 have been suggested in the literature. The details of these approaches are outside the scope of this text. Interested readers are referred to Scales [24].

4.5 OBSERVATION STRATEGY FOR ROBOT KINEMATIC IDENTIFICATION

Observation strategy for calibration refers to the selection of robot configurations and the number of observations to be made during the calibration experiment. The selection of measurement configurations during robot calibration plays an important role in determining the accuracy and speed of convergence of the least-squares identification algorithms. The kinematic parameter errors are not all equally observable. The "visibility" of each unknown parameter varies from one robot configuration to another. There may even be configurations in which some of the kinematic parameters are not observable at all.

Paying close attention to the observability issue and planning the measurement strategy accordingly may save on the total number of different robot configurations that are needed to be attained to obtain an accurate identification. It is further argued [4, 21] that there exist robot configurations at which the impact of geometric errors on the total accuracy overshadows the accuracy error attributed to nongeometric errors. Thus, by proper selection of measurement configurations the effects of unmodeled errors on the identification of the geometric parameters may become less significant.

The synthesis of an "optimal" observation strategy for a given manipulator in the presence of noise and uncertainty is as yet an unanswered research issue. The procedures described in this section could help in deciding the observation strategy for given robots while at the same time providing a measure of confidence in the results of the parameter identification.

For a given robot and measurement method we start by estimating the accuracy of the measurement system, the measurement noise, the resolution and uncertainty for the robot encoders, and the approximate range of motion of the robot joints during the observations. We then do a series of simulations tabulating the effects of strategy, number of observations, and joint range of motion on the accuracy of identification. We also note measures such as the condition number or the observability index of the identification Jacobian for each case.

The range of measures observed will depend on the details of the particular calibration problem. The units used for lengths and angles, the dimensions of the manipulator, the type of measurement method, and the scaling of the rows and columns of the Jacobian will all affect the range of measures observed. The simulations done before the experiment will help to know what range of measures apply to the problem being addressed.

The simulations should show if the proposed experimental setup does have the potential to identify the robot parameters to the accuracy desired. If necessary, the experimental method may be modified at this stage itself based on the results of the simulation. For example, the offset of the tool from the wrist of a robot may need to be increased so as to improve the observability of the wrist parameters. We then conduct the actual calibration experiment using the best observation strategy permitted by the constraints of the measurement, and a number of observations decided from the results of the simulation tables. We compute the measures for the experiment using the actual robot configurations of the observations and the nominal kinematics. This experimental condition number may now be used to refer back to the simulation tables and estimate the quality of the observation strategy during the experiment, and suggest a level of confidence in the parameters identified.

Two groups of references provide the literature background for this section. The first approach [7] focuses on the familiar numerical analysis concept of "condition number" of a matrix. Such a number provides invaluable information regarding the sensitivity of the least-squares identification algorithm to modeling errors and noise in the measured data. The key ideas are described in detail in Sections 4.5.1 and 4.5.2.

The second approach [21] adopts an observability index as a performance measure. There are similarities to the condition number of the previous section. The method is described in Section 4.5.3. Section 4.5.4 provides more insight into the question of useful and informative measurement configurations using the Kalman filtering formulation.

4.5.1 Numerical Sensitivity of Least-Squares Identification Algorithms

The solution of a linear least-squares identification problem in general amounts to the solution of an algebraic equation

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.157}$$

where $\mathbf{A}$ is a nonsingular matrix of order n. The vector $\mathbf{b}$ corresponds to the measured data and as such is subject to uncertainty. The matrix $\mathbf{A}$ is related to the measurement configuration and the system model and may itself be subject to errors. Thus:

$$\mathbf{x} + \delta\mathbf{x} = (\mathbf{A} + \delta\mathbf{A})^{-1}(\mathbf{b} + \delta\mathbf{b}) \tag{4.158}$$

where the solution error $\delta\mathbf{x}$ reflects the sensitivity of the identification algorithm to errors in the model and the data characterized in Equation 4.158 as $\delta\mathbf{A}$ and $\delta\mathbf{b}$.

As is well known to numerical analysts [11] and as shown next, such sensitivity may be characterized in terms of a single number derived from the matrix $\mathbf{A}$.

We start with a brief review of the "norm" concept. The norm of a vector $\mathbf{x}$, denoted as $\|\mathbf{x}\|$ is a function that assigns a real number to any vector $\mathbf{x}$ and

has the following three consistency properties:

1. $\|c\mathbf{x}\| = |c|\,\|\mathbf{x}\|$ for all real c and all vectors $\mathbf{x}$.
2. $\|\mathbf{x}\| > 0$ for all $\mathbf{x} \neq 0$, and $\|\mathbf{x}\| = 0$ only for $\mathbf{x} = 0$.
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for all $\mathbf{x}$ and $\mathbf{y}$ (The triangle inequality).

In particular, three of the most common norms that are used for n-dimensional vectors $\mathbf{x}$ with real elements are the 1-norm, 2-norm, and infinity-norm as follows:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{4.159}$$

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2} \tag{4.160}$$

$$\|\mathbf{x}\|_\infty = \max |x_i| \qquad \text{for } 1 \leq i \leq n \tag{4.161}$$

where the 2-norm is often referred to as the Euclidean norm.

An induced matrix norm is defined as follows:

$$\|\mathbf{A}\| = \max \left\{ \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \qquad \text{for } \mathbf{x} \neq 0 \right\} \tag{4.162}$$

Thus, an induced matrix norm depends on the vector norm that is used. One can show that using the 1-norm produces the maximum column norm for the induced matrix norm, as given by

$$\|\mathbf{A}\|_1 = \max \left\{ \sum_{i=1}^{n} |a_{ij}| \qquad \text{for } 1 \leq j \leq n \right\} \tag{4.163}$$

Use of the vector 2-norm gives the largest singular value of the matrix norm of $\mathbf{A}$ which is also the largest eigenvalue of $\mathbf{A}\mathbf{A}^T$ and use of the ∞-norm yields the maximum row sum as given by

$$\|\mathbf{A}\|_\infty = \max \left\{ \sum_{j=1}^{n} |a_{ij}| \qquad \text{for } 1 \leq i \leq n \right\} \tag{4.164}$$

A well known inequality known as Schwartz inequality is as follows:

$$\|\mathbf{A}\mathbf{B}\| \leq \|\mathbf{A}\|\,\|\mathbf{B}\| \tag{4.165}$$

We are now ready to start with the sensitivity analysis. Assume first that $\delta\mathbf{A} = 0$ but $\delta\mathbf{b} \neq 0$. Then

$$\|\delta\mathbf{x}\| \leq \|\mathbf{A}^{-1}\|\,\|\delta\mathbf{b}\| \tag{4.166}$$

where equality is possible for certain vectors $\delta\mathbf{b}$.

Similar to Equation 4.166 one may write:

$$\|\mathbf{b}\| \leq \|\mathbf{A}\| \, \|\mathbf{x}\| \tag{4.167}$$

Combining Equations 4.166 and 4.167 and assuming that $\mathbf{b} \neq 0$, we find that

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\| \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \tag{4.168}$$

where equality can occur and, therefore, the bound is tight.

Define a condition number $\kappa(\mathbf{A})$ as follows

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\| \tag{4.169}$$

The larger the condition number the larger is the numerical sensitivity of the solution to data errors.

Ideally $\kappa(\mathbf{A}) = 1$ which is the lowest bound on $\kappa(\mathbf{A})$. The condition number depends on the particular vector norm that is used. For instance for the Euclidean norm we find that $\kappa(\mathbf{A})$ is simply the largest singular value of $\mathbf{A}$ divided by the smallest singular value of $\mathbf{A}$. Assuming next that $\delta\mathbf{A} \neq 0$ but $\delta\mathbf{b} = 0$, interestingly enough, the same condition number turns up as follows:

$$\delta\mathbf{x} = [(\mathbf{A} + \delta\mathbf{A})^{-1} - \mathbf{A}^{-1}]\mathbf{b} \tag{4.170}$$

Using now the matrix identity

$$\mathbf{B}^{-1} - \mathbf{A}^{-1} = \mathbf{A}^{-1}(\mathbf{A} - \mathbf{B})\mathbf{B}^{-1} \tag{4.171}$$

One gets

$$\|\delta\mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \, \|\delta\mathbf{A}\| \, \|\mathbf{x} + \delta\mathbf{x}\| \tag{4.172}$$

or

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x} + \delta\mathbf{x}\|} \leq \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\| \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} \tag{4.173}$$

Linear least-squares routines involve the solution of an overdetermined system $\mathbf{Ax} = \mathbf{b}$ such that $\mathbf{A}$ is an $m \times n$ matrix where $m \geq n$. The least-squares solution employs $(\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{A}^T\mathbf{b}$. The condition number for the overdetermined system is the condition number of the $n \times n$ symmetric matrix $\mathbf{C} = \mathbf{A}^T\mathbf{A}$. The robot calibration identification equation $\delta\mathbf{T} = \mathbf{J}\delta\mathbf{k}$ is an overdetermined system. The condition number for the identification Jacobian is thus given by

$$\kappa(\mathbf{J}) = \|(\mathbf{J}^T\mathbf{J})\| \, \|(\mathbf{J}^T\mathbf{J})^{-1}\| \tag{4.174}$$

These concepts may be best illustrated through the example that is presented in the following section.

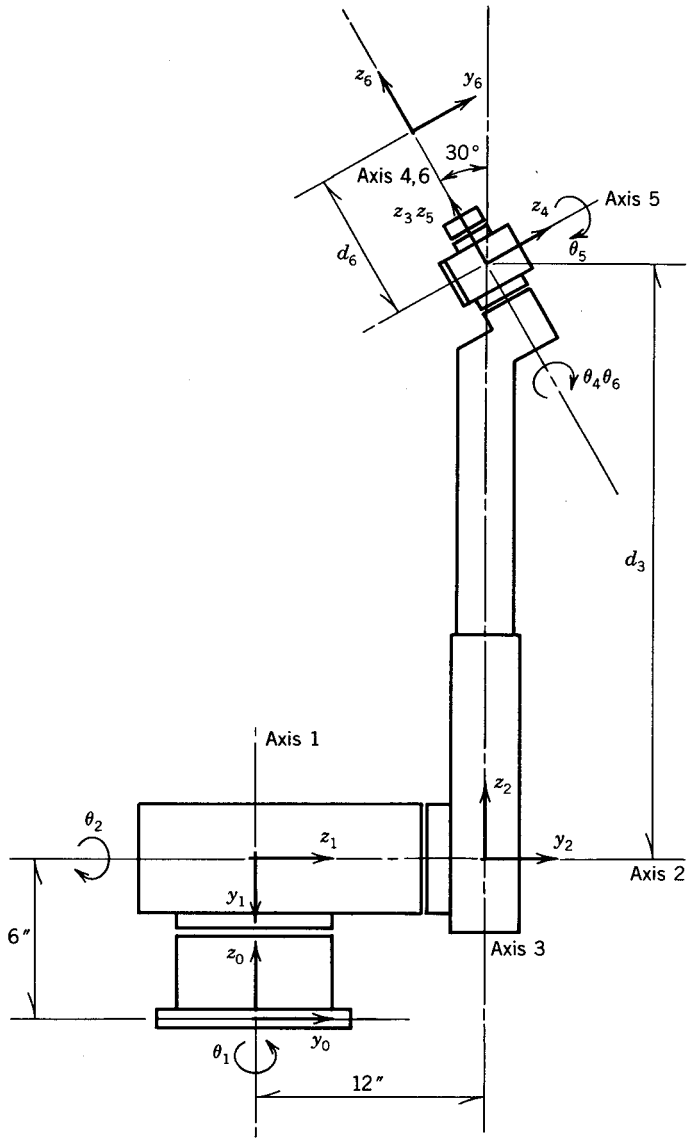


Figure 4.3. The 5R1P manipulator.

4.5.2 Identification Study of a 5R1P Manipulator

The 5R1P manipulator arm is a six-degree-of-freedom manipulator arm with five revolute joints and one prismatic joint. The manipulator is shown in Figure 4.3. The link parameters for the 5R1P manipulator are shown in Table 4.1.

The lengths are in inches and the angles are in degrees. The manipulator has a working volume of about 27 ft^3. There are 24 link parameters: θ_i, r_i, l_i, α_i, for

TABLE 4.1. Link Parameters for the 5R1P Manipulator

Link i	α_i (°)	l_i (in.)	r_i (in.)	θ_i
1	−90	0	6	θ_1
2	+90	0	12	θ_2
3	+30	0	r_3	0
4	−90	0	0	θ_4
5	+90	0	0	θ_5
6	0	l_6	r_6	θ_6

$i = 1, 2, \ldots, 6$. The $\delta\theta_i$s for $i = 1, 2, 4, 5, 6$ will be the encoder offsets at the revolute joints. The δr_3 will be the encoder offset for the prismatic joint. α_3 is chosen to be 30° instead of 0° so that axis 2 and axis 3 will not to be parallel. This permits using the DH notation for the calibration model. Note that this choice of α_3 does not in any way reduce the degrees-of-freedom for the 5R1P manipulator arm. We assume that the measurement method is capable of locating the exact *position* of the robot end effector in the base frame coordinates. Thus, we are not simulating the measurement of end effector orientation. The measurement is a partial measurement, three out of the possible six degrees-of-freedom of end effector pose. This choice of measurement method does have consequences on the robot parameter identification. One immediate consequence is that the last link twist α_6 cannot be identified since it will never affect the position of the end effector. We proceed by assuming that $\alpha_6 = 0.0$ is a constant.

Considerations of model completeness (see Chapter 2 and Section 4.7) dictate that a 5R1P robot manipulator may have a maximum of 22 kinematic parameters (excluding the 6 parameters required to locate the world frame): 4 parameters per revolute joint and 2 parameters for the prismatic joint. We have 24 parameters in Table 4.1. It is not immediately obvious which two parameters in Table 4.1 are to be defined as constants. Review of the definition of the DH notation for prismatic joints [23] shows that it is l_i and r_{i+1} that must be set to a value of 0 when joint i is prismatic. In the case of the 5R1P manipulator, a_3 and d_4 are constant and equal to 0. This is because the location of frame 3 is determined by the common normal between axes 4 and 5. Since the axis of the prismatic joint (axis 3) is not fixed in its location in space, it is free to be moved so that it goes through the origin of frame 3 exactly. This implies that l_3 and r_4 are always 0.

The flowchart for the calibration simulations is shown in Figure 4.4. The approach taken during the course of the identification simulations was as follows. The number of observations and the joint motion range for a simulation were decided. Joint sets of measurement configurations were generated by using one of four strategies. The 5R1P manipulator forward kinematic model was then applied to these joint sets to generate the observation data. The input to this program was the link parameter table for the actual 5R1P robot manipulator to be identified. The output data file contained the joint values and the position coordinates of the end effector at each observation. The random noise of measurement and the encoder noise were superimposed on the position measure-

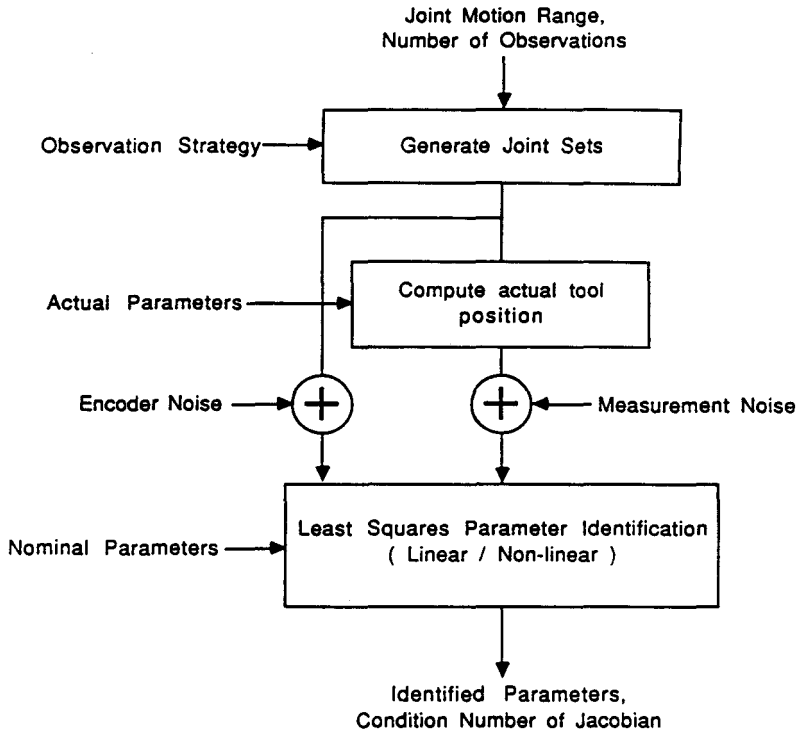


Figure 4.4. Flowchart for calibration simulations.

ments and joint values before being saved to the data file. The input to the identification program consisted of the nominal link parameter table and the observation data file. The programs to implement the linear and nonlinear identification methods were written in FORTRAN so as to be able to call the IMSL Mathematical Subroutine Library routines. These routines provide powerful numerical identification software. The simulations were run on a Digital VAX 8300 running VMS 4.7.

The selection of a particular manipulator (5R1P) for simulation determines the number of links and the types of joints in the kinematic model. Recall that for the manipulator chosen we are attempting to identify a total of 21 kinematic parameters ($l_3 = 0.0$ and $r_4 = 0.0$ by definition for joint 3 prismatic, $\alpha_6 = 0.0$ for the last link frame). Some of the factors that can be expected to have an effect on robot parameter identification are (1) choice of the kinematic model (for instance, screw matrix notation instead of the DH notation), (2) type of measurement method for observations (position of end effector, orientation of end effector, line of sight to end effector), (3) number of measurements, (4) accuracy of the observation measurements, (5) noise in measurements, (6) accuracy and noise at the robot joint encoders, (7) quality of the initial estimate of parameters, (8) observation strategy, that is the selection of the robot configurations during measurements

(joint angles, joint distance), (9) range of motion of the robot joints during calibration (as a consequence of the constraints of a particular measurement set-up), and (10) type of identification parameters (geometric, nongeometric). The performance of the identification methods is judged by (1) success of convergence, (2) accuracy of the identification, and (3) speed of processing.

Tables 4.2 to 4.8 show the results of simulations aimed at determining the effect of the above factors on the performance of identification by the linear and nonlinear methods. The symbols used in the tables are as follows. The variables for the simulations were $\{l_6, r_6\}$, $\{\delta\theta, \delta\alpha\}$, $\{\delta r, \delta l\}$, n_{digits}, $\{\varepsilon_m/2\}$, $\{\varepsilon_e/2\}$, S, m, and R. The results for the linear least-squares method were: n_{iter}, t_{CPU}, δ_{rms}, and $\kappa(\mathbf{J})$. The results for the nonlinear least-squares method were n_{iter}, n_{eval}, t_{CPU}, and δ_{rms}. These symbols are explained below.

The magnitude of the last link length and last joint offset is "l_6, r_6" (inches). This corresponds to the tool distance from the wrist. The error value used for all the θ_i and α_i parameters in the robot model is "$\delta\theta, \delta\alpha$" (degrees). The error value used for all r_i and l_i parameters in the robot model is "$\delta r, \delta l$" (inches). The above two variables are a measure of the distance of the parameter guess from the actual parameters. The routines assume nominal parameters such that these error values are guessed to be zero when starting the identification. Thus, larger values of error parameters above correspond to a worse guess. The number of decimal digits to which the observation measurements are stored in the data file is "n_{digits}." This corresponds to the accuracy of the measurement system. The half width on the random noise in measurement is "$\varepsilon_m/2$" (inches). The half width on the encoder noise is "$\varepsilon_e/2$" (degrees/inches). In both cases, the noise is assumed to have a uniform distribution with zero mean. The observation strategy type is "S." The simulations were done using four observation strategies. These strategies are numbered from 1 to 4. The number of observations is "m." The joint range of motion is "R." The simulations were done using five joint ranges. These are denoted by N, F, H, Q, and E. The notation stands for "normal, full, half, quarter, and eighth." These variables are explained in the discussion of the tables below.

The number of iterations that the identification ran before convergence (LSQRR as well as UNLSF) is "n_{iter}." The number of function evaluations during identification (UNLSF) is "n_{eval}." The charged CPU time (in min:sec) for execution on a VAX 8300 is "t_{CPU}." The root mean square deviation in the identified parameters from the actual error values which were set to $\delta\theta$, $\delta\alpha$, δr, and δl above is "δ_{rms}." This result is a measure of the accuracy of the identification. The condition number of the identification Jacobian is "$\kappa(\mathbf{J})$." The significance of this condition number is discussed later in this section.

The parameters identified by the two methods (linear and nonlinear) under the same conditions were always *identical*. Tables 4.2 to 4.7 have only one set of results (LSQRR) because the identification results were identical for the two methods every time. Table 4.8 shows the comparison of the two methods.

Table 4.2 shows the effect of tool distance from the robot wrist. This result is a consequence of the fact that we are simulating the measurement of tool position

TABLE 4.2. Effect of Tool Distance from Wrist

Variable	Results				
			δ_{rms}		
l_6, r_6	n_{iter}	t_{CPU}	θ, α	r, l	$\kappa(\mathbf{J})$
10.0	5	14	0.00078	0.00017	260
5.0	5	14	0.0016	0.00020	252
1.0	5	14	0.0083	0.00021	373
0.5	5	14	0.017	0.00022	480
0.1	5	18	0.049	0.00028	1848
	7				

Constants							
$\delta\theta, \delta\alpha$	$\delta r, \delta l$	n_{digits}	$\varepsilon_m/2$	$\varepsilon_e/2$	S	m	R
5.0	0.5	4	0.001	0.001	4	36	N

(without orientation). However, any calibration setup that arrives at the end effector orientation from position measurements may exhibit a similar relationship. It was found that the magnitude of the link length l_6 and the offset distance r_6 had a direct bearing on the identification accuracy for the last few angular parameters ($\delta\theta_5$, $\delta\alpha_5$, and $\delta\theta_6$). This is because these angular parameters may affect the position of the end effector only in proportion to the magnitude of the link length and joint offset. Thus l_6 and r_6 represent the offset (distance) of the tool from the wrist of the robot, and this offset should be large enough to provide a "lever arm" for the identification of the wrist joint parameters. This would be common sense during an actual calibration experiment, however the extent of the effect as seen in the simulations might not be obvious. The results show that as the tool distance is reduced from 10 to 0.1 in., the accuracy of identification for the angular parameters reduces from 0.0008 to 0.05°. The accuracy of identification of the distance parameters also reduces slightly. The condition number of the identification Jacobian increased from 260 to 1848. The simulations are for 36 observations (as listed under the "constants" column).

Table 4.3 shows the effect of parameter guess on identification. The quality of the guess had no effect on the accuracy of identification, only on the time required to process the case. The parameter guess was increased from 0.1° and 0.01 in. to as much as 20° and 2 in. The accuracy of identification remained constant. The time for processing increased from 11 to 18 sec.

Table 4.4 shows the effect of measurement accuracy on the identifications. As the number of digits in the measurement is reduced from 8 to 2, the accuracy of identification reduces from 0.0004 to 0.003°, and from 0.0001 to 0.0008 in. For the purpose of this table, the measurements were assumed to be noise free. The remaining constants are the same as usual. The fact that the parameter identification accuracy is directly related to experimental measurement accuracy should

TABLE 4.3. Effect of Parameter Guess

Variables		Results			
				δ_{rms}	
$\delta\theta, \delta\alpha$	$\delta r, \delta l$	n_{iter}	t_{CPU}	θ, α	d, a
0.1	0.01	3	11	0.0008	0.0002
1.0	0.10	4	12	0.0008	0.0002
5.0	0.50	5	14	0.0008	0.0002
10.0	1.00	6	17	0.0008	0.0002
20.0	2.00	7	18	0.0007	0.0002

Constants						
l_6, r_6	n_{digits}	$\varepsilon_m/2$	$\varepsilon_e/2$	S	m	R
10.0	4	0.001	0.001	4	36	N

TABLE 4.4. Effect of Measurement Accuracy

Variable	Results			
			δ_{rms}	
n_{digits}	n_{iter}	t_{CPU}	θ, α	r, l
8	5	15	0.00038	0.00014
6	5	15	0.00039	0.00014
4	5	15	0.00042	0.00015
2	5	15	0.0028	0.00080

Constants							
l_6, r_6	$\delta\theta, \delta\alpha$	$\delta d, \delta$a	$\varepsilon_m/2$	$\varepsilon_e/2$	S	m	R
10.0	5.0	0.5	0.0	0.001	4	36	N

of course be expected. Increasing the number of observations helps to reduce the effect of random noise in measurement, but the basic measurement process must possess an accuracy that is comparable to the parameter identification accuracy desired. In fact, in the presence of measurement noise, the accuracy of identification will begin to approach the accuracy of measurement only after a large number of measurements have been taken.

Table 4.5 shows the effect of measurement and encoder noise. With no noise and four decimal digits in the measurement, the accuracy of identification is 0.00007° and 0.00002 in. With encoder noise of 0.0001, the accuracy reduces to 0.0004° and 0.0002 in. The addition of measurement noise of 0.01 in. reduces the accuracy of identification down to 0.006° and 0.001 in. The table shows that measurement and encoder noise have a substantial effect on the accuracy of identification. The question from the point of view of design of calibration

TABLE 4.5. Effect of Measurement and Encoder Noise

Variables		Results			
				δ_{rms}	
$\varepsilon_m/2$	$\varepsilon_e/2$	n_{iter}	t_{CPU}	θ, α	r, l
0.0	0.0	5	15	0.00007	0.00002
0.0	0.001	5	15	0.0004	0.0002
0.0001	0.001	5	15	0.0004	0.0002
0.001	0.001	5	15	0.0008	0.0002
0.01	0.001	5	15	0.006	0.001

Constants						
l_6, r_6	$\delta\theta, \delta\alpha$	$\delta r, \delta l$	n_{digits}	S	m	R
10.0	5.0	0.5	4	4	36	N

experiments is one of how many observations must be taken to neutralize the effect of such random noise.

Table 4.6 shows the effect of increasing the number of observations using different observation strategies. The strategies refer to the method used for generating the robot configurations at which the observations are made. Strategy number 1 consists of moving each joint by constant increments simultaneously. Strategy 2 consists of moving one joint at a time while the remaining five joints are held constant. This is done for all joints. Strategy 3 consists of holding one joint constant while the rest of the joints are moved by constant increments simultaneously. This is done for all six joints. Strategy 4 is a Monte Carlo type method. The six joint variables are generated from six independent uniformly distributed random variables. The random variables are scaled so as to span the permissible range of each joint.

For each strategy, the number of observations is increased from 36 (or less) to 360. The results show that increasing the number of observations improves the accuracy of identification in all cases. However, some strategies are worse than others. Strategy 1 is an extreme example of such a "bad" strategy. Even with 360 observations, the accuracy of identification for strategy 1 is worse than any of the other strategies at 36 observations. It is possible to see that strategy 1 would be considered a bad strategy in any practical calibration experiment since it would essentially result in observations made along a single space curve in the robot work volume. Strategies 2, 3, and 4 result in better identification, with 4 doing somewhat better than 3, and 3 doing better than 2. The condition number of the identification Jacobian for these strategies shows a difference between Strategy 1 (over 20,000) and the other strategies (less than 1200). The better a strategy, the lower its corresponding condition number. The increase in observations show diminishing returns in terms of improvement in identification accuracy. For Strategy 4 at 12 observations, the accuracy of identification is 0.002°

TABLE 4.6. Effect of Observation Strategy

Variables		Results				
				δ_{rms}		
S	m	n_{iter}	t_{CPU}	θ, α	r, l	$\kappa(\mathbf{J})$
1	7	6	7	0.032	0.013	23020
	36	6	16	0.0078	0.0037	20024
	90	6	33	0.0044	0.0020	21664
	180	6	1:02	0.0047	0.0025	22307
	360	6	2:03	0.0054	0.0019	22650
2	36	5	14	0.0018	0.00028	1203
	90	5	31	0.0010	0.00026	1138
	180	5	55	0.0012	0.00019	1119
	360	5	1:47	0.00045	0.00013	1110
3	36	5	14	0.0013	0.00048	364
	90	5	29	0.00065	0.00021	396
	180	5	55	0.00089	0.00010	420
	360	5	1:44	0.00032	0.000090	434
4	12	5	8	0.0021	0.00029	539
	36	5	14	0.00078	0.00017	260
	90	5	28	0.00079	0.00011	209
	180	5	54	0.00042	0.00011	204
	360	5	1:45	0.00030	0.00010	189

Constants						
l_6, r_6	$\delta\theta, \delta\alpha$	$\delta r, \delta l$	n_{digits}	$\varepsilon_m/2$	$\varepsilon_e/2$	R
10.0	5.0	0.5	4	0.001	0.001	N

and 0.0003 in. At 36 observations, the accuracy is 0.0008° and 0.0002 in. At 360 observations, the accuracy is 0.0003° and 0.0001 in. It should be noted that the accuracy of measurement (0.0001 in.) and the resolution of the joint encoders (0.0001°) assumed in these simulations are very conservative. The noise levels are also very low. An actual calibration experiment may be expected to show much less accuracy of identification.

The simulations of the previous table assume that the calibration measurements can be done in any valid pose of the robot manipulator. An actual robot manipulator has limits on the range of motion of each joint. We call this range "N" (normal) and the simulations of the earlier tables are all based on this range. To simulate the limited range of joint motion that would result in an actual calibration experiment we have the four choices of "F, H, Q, and E." The constraints of the measurement system during a calibration may keep all robot joints from being exercised through their full range. "F" corresponds to the maximum possible full range of motion for all joints. "H" corresponds to half the maximum range for each of the first three positioning joints. Similarly, "Q"

TABLE 4.7. Effect of Joint Range of Motion

R	θ_1	θ_2	r_3	θ_4	θ_5	θ_6
N	$0 \rightarrow 300$	$-120 \rightarrow 120$	$12 \rightarrow 48$	$0 \rightarrow 360$	$-120 \rightarrow 120$	$0 \rightarrow 360$
F	$0 \rightarrow 360$	$-180 \rightarrow 180$	$12 \rightarrow 48$			
H	$0 \rightarrow 180$	$-90 \rightarrow 90$	$30 \rightarrow 48$			
Q	$0 \rightarrow 90$	$-45 \rightarrow 45$	$39 \rightarrow 48$			
E	$0 \rightarrow 45$	$-22 \rightarrow 22$	$44 \rightarrow 48$			

Variables		Results			
			δ_{rms}		
S	R	n_{iter}	θ, α	r, l	$\kappa(\mathbf{J})$
1	F	6	0.015	0.0033	16375
	H	6	0.030	0.016	47802
	Q	8	0.196	0.089	303101
	E	Diverge	—	—	1×10^6
2	F	5	0.0032	0.00044	1260
	H	5	0.0024	0.00045	1393
	Q	5	0.0037	0.0011	3330
	E	5	0.010	0.0027	9260
3	F	5	0.0011	0.00019	282
	H	5	0.0020	0.00069	909
	Q	5	0.0010	0.00041	2579
	E	5	0.0067	0.0023	6640
4	F	5	0.0013	0.00020	239
	H	5	0.0016	0.00076	501
	Q	5	0.0024	0.0017	1696
	E	5	0.0069	0.0041	4804

Constants						
l_6, r_6	$\delta\theta, \delta\alpha$	$\delta r, \delta l$	n_{digits}	$\varepsilon_m/2$	$\varepsilon_e/2$	m
5.0	5.0	0.5	4	0.001	0.001	36

and "E" correspond to quarter and eighth range for the first three joints. The last three joints of the 5R1P manipulator are responsible for orientation of the tool. In the simulations, these joints are always allowed to go through their normal range.

Table 4.7 shows the effect of the joint range of motion for different strategies. For each strategy, the range of joint motion is varied from F to E. In every case, the reduction in joint range of motion results in reduced accuracy of identification. The magnitude of this effect is substantial. For Strategy 4 at F range, the accuracy of identification is 0.001° and 0.0002 in. At E range, the accuracy is 0.007° and 0.004 in. The reduction in range of motion is accompanied by an increase in the condition number of the identification Jacobian. In the case of Strategy 1, the case of range E results in divergence and failure of the identification procedure.

TABLE 4.8. Comparison of Identification Techniques

Variable	Results				
	LSQRR		UNLSF		
m	n_{iter}	t_{CPU}	n_{iter}	n_{eval}	t_{CPU}
36	5	14	5	7	1:22
90	5	28	5	6	3:39
180	5	54	5	6	6:26
360	5	1:45	5	6	13:39

Constants							
l_6, r_6	$\delta\theta, \delta\alpha$	$\delta r, \delta l$	n_{digits}	$\varepsilon_m/2$	$\varepsilon_e/2$	S	R
10.0	5.0	0.5	4	0.001	0.001	4	N

Table 4.8 shows the comparison of the two identification techniques. As mentioned earlier, both linear and nonlinear methods give identical results under the same conditions. This table takes four cases of increasing number of observations and compares the time taken by the two methods. The linear method is four to eight times faster than the nonlinear method. At 360 observations, the linear method takes less than 2 min, while the nonlinear method takes about 13 min of CPU time.

The condition number of the identification Jacobian is an indicator of the observability of the parameters to be identified. Decreasing the tool offset reduces the observability of the wrist angular parameters and this is accompanied by increase in the condition numbers in Table 4.2. A bad observation strategy such as Strategy 1 is marked by very large condition numbers in Table 4.6. Reduction in the range of motion of a joint during calibration implies lesser observability of the parameters of that joint and this is accompanied by order of magnitude increase in the condition numbers in Table 4.7.

Both the linear and nonlinear least squares are found to give identical parameter estimates for all of the cases considered. The linear method is as much as eight times faster than the nonlinear method. However, the linear least-squares method requires the user to write a program to compute the elements of the identification Jacobian for the manipulator and model being used. This is a nontrivial task that requires substantial work and may be a source of potential programming errors. By comparison, the nonlinear method requires that the user supply only the subroutine to compute the forward solution for the manipulator to be identified. The method is robust in the face of large parameter uncertainties. It may be easily modified to handle different kinematic models (including those with non-geometric parameters), and may be extended to problems where it is required to identify not only the manipulator kinematics but also calibrate the location of a world frame and other frames in the robot workspace. The longest time for identification by this method in our simulations was about 15 min of CPU time. Since calibration is to be done off-line, this is not an unreasonable duration to wait.

The simulations showed that quality of the initial parameter estimates does not affect identification accuracy and need not be among the important concerns. On the other hand, accuracy of parameter identification is influenced by the accuracy and noise in measurements, encoder resolution and uncertainty, selection of measurement configurations, number of observations, and joint range of motion. It is seen that the condition number of the identification Jacobian during calibration is an indicator of the observability of the parameters. Identification is aided by providing an adequate distance of the tool from the robot wrist during calibration. In the presence of noise and uncertainty, an adequately large number of observations using a good observation strategy is essential. The question of *how many* observations are adequate during a particular calibration experiment is not obvious, and the synthesis of an "optimal" observation strategy is an unanswered research issue at this time.

4.5.3 Observability of Kinematic Parameter Errors

One of the basic and most important tools of modern numerical analysis, particularly numerical linear algebra, is the singular value decomposition (SVD). This section is based on several references by Menq and Borm [4, 21] in which the observability of parameter error is studied via the use of the SVD method based on the linearized error model and measured position data. Subsequently, an observability index is defined and is used as a measure of observability of a set of measurement configurations in simulation studies similar to those done in the previous section.

We shall start with a brief review of SVD. The reader is referred to the work by Klema and Laub [15] for more details.

Let $\mathbf{A}$ be any $m \times n$ matrix with real elements such that $n \geq m$. Then there exists an $m \times m$ orthogonal matrix $\mathbf{U}$ and an $n \times n$ orthogonal matrix $\mathbf{V}$ such that

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{4.175}$$

where

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & & 0 & 0 & 0 & 0 \\ 0 & 0 & & & & & \\ \vdots & & & & & & 0 \\ 0 & --- & & \lambda_m & 0 & \cdots & 0 \end{bmatrix} \tag{4.176}$$

with

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > \lambda_{r+1} = \cdots = \lambda_n = 0 \tag{4.177}$$

The numbers $\{\lambda_1, \ldots, \lambda_n\}$ are the singular values of $\mathbf{A}$, that is, the positive square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ and r is the rank of $\mathbf{A}$.

The columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^T$ while the columns of $\mathbf{V}$ are eigenvectors of $\mathbf{A}^T\mathbf{A}$, that correspond to the eigenvalues $\lambda_1^2, \ldots, \lambda_n^2$ of $\mathbf{A}^T\mathbf{A}$ arranged in an ascending order. The columns of $\mathbf{U}$ form a set of orthonormal vectors, and so do the columns of $\mathbf{V}$. Quite a few software packages exist that compute SVD very efficiently. For example, see Klema and Laub [15].

To see how all this applies to the kinematic parameter error identification recall Equation 4.28 for the linearized error model in vector form:

$$\mathbf{e}(j) = \mathbf{H}(j)\mathbf{X}, \qquad j = 1, \ldots, m \tag{4.178}$$

where $\mathbf{X}$ is the vector of unknown kinematic parameter errors and $\mathbf{e}(j)$ is the difference between the measured pose and the estimated pose based on the robot nominal kinematics. $\mathbf{H}(j)$ is derived from the identification Jacobian at the measurement configuration j. There are m measurement configurations.

Let $\mathbf{E}$ be the aggregated vector of pose errors in all m measurement configurations:

$$\mathbf{E} = \begin{pmatrix} \mathbf{e}(1) \\ \vdots \\ \mathbf{e}(m) \end{pmatrix} \tag{4.179}$$

Then

$$\mathbf{E} = \mathbf{H}\mathbf{X} \tag{4.180}$$

where $\mathbf{H}$ aggregates the matrices $\mathbf{H}(j), j = 1, \ldots, m$, as in Equation 4.35.

Singularities in the identification Jacobian may arise depending on the selected kinematic modeling convention. A common problem is that of $\mathbf{X}$ containing elements that are indistinguishable at any robot configuration because of the kinematic nature of the robot. An ill-posed identification problem may become well posed on appropriate reduction of $\mathbf{X}$. It is assumed throughout that Equation 4.180 represents a well-posed problem.

Singular value decomposition may now be applied either to each matrix $\mathbf{H}(j)$ (Equation 4.178) of each measurement configuration or to the aggregated matrix $\mathbf{H}$ of Equation 4.180. Thus, assuming that $\mathbf{X}$ is l-dimensional and therefore $\mathbf{H}(j)$ is $6 \times l$:

$$\mathbf{H}(j) = \mathbf{U}_j \mathbf{\Lambda}_j \mathbf{V}_j^T \tag{4.181}$$

where

$$\mathbf{U}_j = [\hat{\mathbf{u}}_{1j}, \hat{\mathbf{u}}_{2j}, \ldots, \hat{\mathbf{u}}_{6j}] \tag{4.182}$$

and $\hat{\mathbf{u}}_{ij}$, $i = 1, \ldots, 6$ are orthonormal 6×1 eigenvectors of $\mathbf{H}(j)\mathbf{H}^T(j)$.

$$\mathbf{V}_j = [\hat{\mathbf{v}}_{1j}, \ldots, \hat{\mathbf{v}}_{lj}] \tag{4.183}$$

$\hat{\mathbf{v}}_{ij}$, $i = 1, \ldots, l$ are orthonormal $l \times 1$ eigenvectors of $\mathbf{H}(j)^T\mathbf{H}(j)$.

All these eigenvectors correspond to the set of eigenvalues $\lambda_1^2 \geq \cdots \geq \lambda_6^2 \geq 0$ of $\mathbf{H}^T(j)\mathbf{H}(j)$.

The error model (Equation 4.178) for each robot configuration j may now be written as follows

$$\mathbf{e}(j) = \sum_{i=1}^{6} \lambda_{ij}(\hat{\mathbf{v}}_{ij}^T\mathbf{X})\mathbf{u}_{ij} \tag{4.184}$$

It is seen that only the projection of $\mathbf{X}$ on the subspace spanned by $\{\mathbf{v}_{1j}, \ldots, \mathbf{v}_{6j}\}$ is transmitted to the position error $\mathbf{e}(j)$. In general $l \geq 6$. Consequently, only a portion of $\mathbf{X}$ can be observed from the measured position error in a single measurement configuration.

Applying the same idea to Equation 4.180 that corresponds to the aggregated errors of m measurement configurations one gets

$$\mathbf{E} = \mathbf{U}\Lambda\mathbf{V}^T\mathbf{X} \tag{4.185}$$

where

$$\mathbf{U} = [\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_{6m}] \tag{4.186}$$

$\{\hat{\mathbf{u}}_i, i = 1, \ldots, 6m\}$ are orthonormal $6m \times 1$ vectors. Likewise

$$\mathbf{V} = [\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_l] \tag{4.187}$$

where $\hat{v}_1$, $i = 1, \ldots, l$ are orthonormal $l \times 1$ vectors.

It is assumed that $6m \geq l$. Thus

$$\Lambda = [\Sigma_1^T, \Sigma_2^T] \tag{4.188}$$

where

$$\Sigma_1 = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_l) \tag{4.189}$$

with $\sigma_1^2 \geq \sigma_1^2 \geq \cdots \geq \sigma_l^2$ being the eigenvalues of $\mathbf{H}^T\mathbf{H}$, and

$$\Sigma_2 = 0 \tag{4.190}$$

Then

$$\mathbf{E} \simeq \sum_{i=1}^{l} \sigma_i(\hat{\mathbf{v}}_i^T\mathbf{X})\hat{\mathbf{u}}_i \tag{4.191}$$

If none of the singular values σ_i is zero, we say that the kinematic parameter error vector $\mathbf{X}$ is observable from the measurement vector $\mathbf{E}$.

It can also be shown from Equation 4.191 that $\|\mathbf{E}\|_2$ is bounded by

$$\sigma_l \|\mathbf{X}\|_2 \leq \|\mathbf{E}\|_2 \leq \sigma_1 \|\mathbf{X}\|_2 \tag{4.192}$$

In the work by Menq and Borm [21] a measure of the observability of $\mathbf{H}$ is thus defined using all singular values $\sigma_1, \ldots, \sigma_l$ as follows:

$$O_{\mathrm{H}} = \frac{(\sigma_1 \sigma_2 \ldots \sigma_l)^{1/l}}{\sqrt{m}} = \frac{[\det(\mathbf{H}^T\mathbf{H})^{1/2l}}{\sqrt{m}} \tag{4.193}$$

The larger the value of $\det(\mathbf{H}^T\mathbf{H})$ the more "observable" $\mathbf{X}$ is given the m measurement configurations. $\mathbf{X}$ becomes unobservable if at least one of the singular values σ_i equals zero.

The same type of observability measure may be defined directly from either the indentification Jacobian or the aggregation of identification Jacobians:

$$O_J = \frac{[\det(\mathbf{J}^T\mathbf{J})]^{1/2l}}{\sqrt{m}} \tag{4.194}$$

where again l is the number of unknown kinematic error parameters. Note that the condition number $\kappa(J)$ defined in the previous section decreases as the observability index increases since it is inversely proportional to the determinant of $\mathbf{J}^T\mathbf{J}$.

4.5.4 Observation Strategy from a Kalman Filtering View

As a preliminary, a time invariant Kalman filtering formulation of the calibration identification phase will be presented. The "time-invariant" formulation of the Kalman filter equations for the identification of $\mathbf{X}$, the vector of kinematic error parameters, is a combination of batch and recursive estimation ideas. Its main purpose is to facilitate the analytic treatment of questions raised related to observation strategy selection.

The idea is to assume that there are k_r repeated measurement readings at each measurement location corresponding to a particular robot pose. $k_r = 1$ means "no repetitions."

Let k_p denote the number of different robot measurement configurations. Obviously the total number of measurements m is given by

$$m = k_p k_r \tag{4.195}$$

Since the order of processing of the measurement data is arbitrary the following particular ordering of measurements is selected:

Measurements taken at pose 1:

$$\mathbf{z}(1), \mathbf{z}(k_p + 1), \ldots, \mathbf{z}[(k_r - 1)k_p + 1]$$

Measurements taken at pose 2:

$$\mathbf{z}(2), \mathbf{z}(k_p + 2), \ldots, \mathbf{z}[(k_r - 1)k_p + 2]$$

Measurements taken at pose k_p:

$$\mathbf{z}(k_p), \mathbf{z}(2k_p), \ldots, \mathbf{z}(k_r k_p)$$

Every group of consecutive k_p measurements may be concatenated to form a measurement vector $\mathbf{\Pi}(i)$.

$$\mathbf{\Pi}(i) = \begin{bmatrix} \mathbf{z}[(i-1)k_p + 1] \\ \mathbf{z}[(i-1)k_p + 2] \\ \vdots \\ \mathbf{z}(ik_p) \end{bmatrix}; \qquad i = 1, \ldots, k_r \tag{4.196}$$

This particular ordering of measurements makes the measurement Equation 4.76 a *periodic* time-varying equation in which the period of $\mathbf{H}(j)$ is k_p, where $j = 1, \ldots, k_r k_p$.

Equivalently the measurement vector $\mathbf{\Pi}(i)$ of Equation 4.196 defines a time-invariant measurement equation

$$\mathbf{\Pi}(i) = \mathbf{\Lambda}\mathbf{X}(i) + \mathbf{\Psi}(i); \qquad i = 1, \ldots, k_r \tag{4.197}$$

where

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{H}(1) \\ \mathbf{H}(2) \\ \vdots \\ \mathbf{H}(k_p) \end{bmatrix} \tag{4.198}$$

$$\mathbf{\Psi}(i) = \begin{bmatrix} \mathbf{v}[(i-1)k_p + 1] \\ \mathbf{v}[(i-1)k_p + 2] \\ \vdots \\ \mathbf{v}(ik_p) \end{bmatrix} \tag{4.199}$$

At a given robot measurement configuration the values of the noise vectors at every repeated measurement vary, however, it is reasonable to assume that the probability distribution of the noise remains unchanged. Thus, denoting the measurement noise covariance matrix at a given configuration j by $\mathbf{\Sigma}_v(j)$, then

$$\mathbf{\Sigma}_v(j) = \mathbf{\Sigma}_v(j + Kk_p); \qquad K = 0, \ldots, k_r - 1 \tag{4.200}$$

Therefore the covariance matrix of $\Psi(i)$ is block diagonal as follows:

$$\Sigma_{\Psi}(i) = \begin{bmatrix} \Sigma_v(1) & 0 & \dots & 0 \\ 0 & \Sigma_v(2) & \dots & 0 \\ \vdots & \vdots & \dots & 0 \\ 0 & 0 & \dots & \Sigma_v(k_p) \end{bmatrix} \tag{4.201}$$

In simple words the above time-invariant model is no other than the batch-processing model with repetitions.

Repeated use of the error covariance Equation 4.70 as i increases yields

$$\Sigma_e(i) = (\Sigma_x^{-1} + i\Lambda^T\Sigma_{\Psi}^{-1}\Lambda)^{-1} \tag{4.202}$$

where $\Sigma_e(i)$ is the error covariance in estimating $\mathbf{X}$ at every repeated measurement i, $i = 1, \dots, k_r$. Therefore if k_p is sufficiently large and the matrix $\Lambda^T\Sigma_{\Psi}^{-1}\Lambda$ is nonsingular, then theoretically as the number of repetitions becomes high, $k_r \to \infty$, the calibration accuracy becomes perfect. This agrees with what has been said before analyzing the no process noise case in Kalman filters.

The requirement that k_p be sufficiently large corresponds to an observability requirement of the system given in Equation 4.197 together with the "process equation":

$$\mathbf{X}(i) = \mathbf{X}(i-1) = \mathbf{X} \tag{4.203}$$

Thus, as the observability of the time-invariant system depends on Λ alone, one requires

$$\text{rank}(\Lambda) = \dim(\mathbf{X}) \tag{4.204}$$

Equation 4.34 implies a necessary condition that k_p should satisfy, namely

$$k_p \geq 1 + \text{int}\left\{\frac{\dim(\mathbf{X})}{\dim[\mathbf{z}(j)]}\right\} \tag{4.205}$$

where int denotes the "largest integer not greater than."

The condition is not sufficient due to the possibility that certain measurement configurations $\mathbf{H}(j)$ do not contribute to the rank of the overall measurement matrix Λ.

If the system given in Equations 4.197 and 4.203 is not fully observable, it simply means that not all elements of the unknown vector $\mathbf{X}$ relate to the information contained in the measurement data $\Pi(i)$; $i = 1, \dots, k_r$. A trivial case of unobservability is when there is a lack of sufficient number of measurement constraint equations as occurs whenever the requirement in Equation 4.205 is violated.

4.5.5 Two-Link Manipulator Example

Consider the two-link planar manipulator problem. It is assumed that the only geometric errors are in the link lengths r_1, r_2 and in the joint variables θ_1, θ_2. The constant errors are denoted as dr_1, dr_2, $d\theta_1$, $d\theta_2$, respectively. The nominal and actual kinematic equations relating the world coordinates (P_x, P_y) to the joint coordinates (θ_1, θ_2) are given as follows.

Nominal Model (omitting the index j from P_{x_0}, P_{y_0}, θ_1 and θ_2):

$$\mathbf{P}_{x_0} = r_1 \cos\theta_1 + r_2 \cos(\theta_1 + \theta_2) \tag{4.206}$$

$$\mathbf{P}_{y_0} = r_1 \sin\theta_1 + r_2 \sin(\theta_1 + \theta_2) \tag{4.207}$$

Actual Model:

$$P_x = (r_1 + dr_1)\cos(\theta_1 + d\theta_1) + (r_2 + dr_2)\cos(\theta_1 + d\theta_1 + \theta_2 + d\theta_2) \tag{4.208}$$

$$P_y = (r_1 + dr_1)\sin(\theta_1 + d\theta_1) + (r_2 + dr_2)\sin(\theta_1 + d\theta_1 + \theta_2 + d\theta_2) \tag{4.209}$$

The resulting linearized measurement equation becomes

$$\mathbf{z}(j) = \mathbf{H}(j)\mathbf{X} + \mathbf{v}(j) \tag{4.210}$$

where

$$\mathbf{z}(j) = \begin{bmatrix} P_x(j) - P_{x_0}(j) + v_x(j) \\ P_y(j) - P_{y_0}(j) + v_y(j) \end{bmatrix} \tag{4.211}$$

and

$$\mathbf{X} = \begin{bmatrix} d\theta_1 \\ d\theta_1 \\ dr_1 \\ dr_2 \end{bmatrix} \tag{4.212}$$

Suppressing the index j from θ_1 and θ_2 and adopting the following shorthand notation:

$$s\theta_i = \sin\theta_i \tag{4.213}$$

$$c\theta_i = \cos\theta_i \tag{4.214}$$

$$s_{ij}\theta = \sin(\theta_i + \theta_j) \tag{4.215}$$

$$c_{ij}\theta = \cos(\theta_i + \theta_j) \tag{4.216}$$

the $\mathbf{H}(j)$ matrix becomes

$$\mathbf{H}(j) = \begin{bmatrix} (-r_1 s\theta_1 - r_2 s_{12}\theta) & -r_2 s_{12}\theta & c\theta_1 & c_{12}\theta \\ (-r_1 s\theta_1 - r_2 c_{12}\theta) & -r_2 c_{12}\theta & s\theta_1 & s_{12}\theta \end{bmatrix} \tag{4.217}$$

Trivially for $k_p = 1$ (one measurement configuration), $\mathbf{\Lambda} = \mathbf{H}(1)$ and is rank defficient and as such the four elements of $\mathbf{X}$ are not uniquely identifiable.

Increasing the number of measurement configurations from 1 to 2, results in having the $\mathbf{\Lambda}$ matrix:

$$\mathbf{\Lambda} = \begin{bmatrix} -r_1 s\theta_1(1) - r_2 s_{12}\theta(1) & -r_2 s_{12}\theta(1) & c\theta_1(1) & c_{12}\theta(1) \\ -r_1 c\theta_1(1) - r_2 c_{12}\theta(1) & -r_2 c_{12}\theta(1) & s\theta_1(1) & s_{12}\theta(1) \\ -r_1 s\theta_1(2) - r_2 s_{12}\theta(2) & -r_2 s_{12}\theta(2) & c\theta_1(2) & c_{12}\theta(2) \\ -r_1 c\theta_1(2) - r_2 c_{12}\theta(2) & -r_2 c_{12}\theta(2) & s\theta_1(2) & s_{12}\theta(2) \end{bmatrix} \tag{4.218}$$

For most pairs of measurement configuration, $\theta_1(1)$, $\theta_2(1)$, $\theta_1(2)$, $\theta_2(2)$, $\mathbf{\Lambda}$ is full rank and the system is observable. Here are a few interesting numerical examples worked out using PC-MATLAB.

1. $r_1 = 2, r_2 = 1$; $\theta_1(1) = 60°$; $\theta_2(1) = 35°$; $\theta_1(2) = 60°$; $\theta_2(2) = 0°$. The resulting rank of $\mathbf{\Lambda}$ is 4. Note that θ_1 is the same in both configurations, and the second configuration is singular. Nevertheless the system is observable.
2. $r_1 = 2, r_2 = 1$; $\theta_1(1) = 73°$; $\theta_2(1) = 0°$; $\theta_1(2) = 34°$; $\theta_2(2) = 180°$. Result: rank $(\mathbf{\Lambda}) = 4$. Here both configurations are singular!
3. $r_1 = 2, r_1 = 1$; $\theta_1(1) = 45°$; $\theta_2(1) = 90°$; $\theta_1(2) = 50°$; $\theta_2(2) = -90°$. Result: rank $(\mathbf{\Lambda}) = 4$. Here there is a "mirror symmetry" between the two configurations.
4. $r_1 = r_2 = 1$; $\theta_1(1) = 14°$; $\theta_2(1) = 120°$; $\theta_1(2) = 134°$; $\theta_2(2) = -120°$. Result: rank $(\mathbf{\Lambda}) = 4$.

This is a particularly interesting case in which both configurations result in the same world coordinates for the end effector. In other words, the total number of calibration fixtures may be smaller than what one expects if contact between the robot end effector and the calibration fixtures may be achieved using a variety of robot configurations.

All of the above examples are illustrated in Figure 4.5. The only case that has been found in which rank $(\mathbf{\Lambda}) < 4$ is the case:

$$\theta_1(1) \neq \theta_1(2); \qquad \theta_2(1) = \theta_2(2) \tag{4.219}$$

In other words, when the robot configurations retain the same shape as shown in Figure 4.6, no unique identification is possible.

The time-invariant identification formulation results from the assumption that k_r repetitions are made at every robotic measurement configuration. The

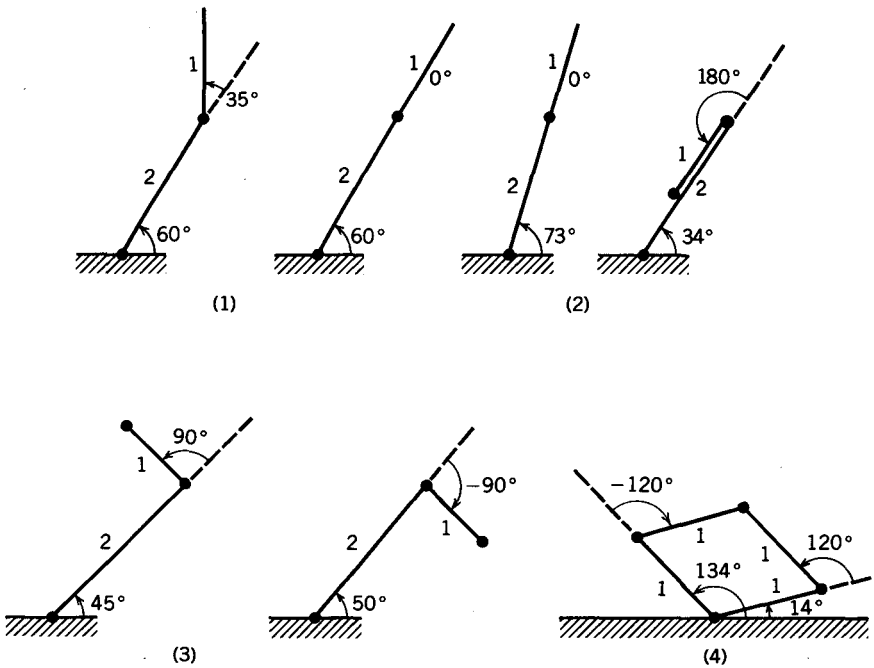


Figure 4.5. Observable configurations for the two link planar manipulator.

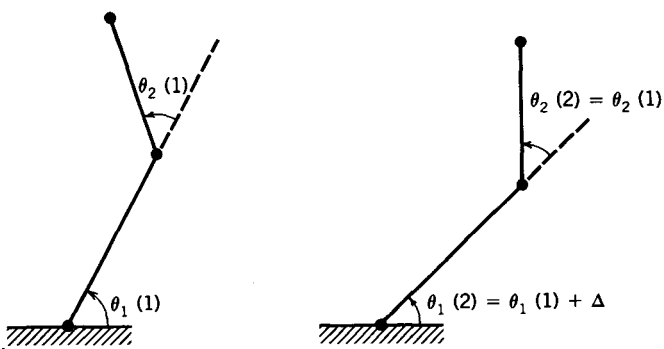


Figure 4.6. Unobservable configurations for the two link planar manipulator.

question that is studied in the next section is that of the necessary minimum number of such repetitions for the sake of getting a low enough identification error in a suitable sense. This question is somewhat artifical and of dubious practical relevance as in most calibration measurement applications there are no repetitions at all, that is $k_r = 1$. Yet, such analysis may provide better insight into the more important problem of necessary number of measurement configurations k_p to achieve a prespecified calibration accuracy.

The basis for analysis are Equations 4.114 and 4.202. The latter relates the error covariance Σ_e to i (the number of repetitions), Σ_x (the initial uncertainty in X), Σ_Ψ (the measurement noise covariance), and Λ (the matrix that depends on the robotic measurement configurations). The former formula is a condition for truncating the repetitions. The termination condition may be absolute as in Equation 4.114, or relative, as follows:

$$\|\mathbf{P}(j)\| \leq \varepsilon \|\mathbf{P}_0\|, \qquad \forall j \geq M, 0 < \varepsilon < 1 \tag{4.220}$$

In other words, when the identification error covariance norm reduces to an acceptable fraction of the initial covariance norm, no more repetitions are needed. The condition above makes sense only for values ε for which

$$\varepsilon \|\mathbf{P}_0\| \geq \|\mathbf{P}_\infty\| \tag{4.221}$$

including the value $\varepsilon = 1$. In the case of a fixed **X** (no process noise) $\mathbf{P}_\infty = 0$ and any $0 < \varepsilon < 1$ may be chosen. By Equation 4.202 $\|\mathbf{P}(j)\|$ as a function of j, $j = 0, 1, 2, \ldots$ is monotonically decreasing.

Equation 4.202 may be rewritten as follows:

$$\mathbf{P}(j) = (\mathbf{I} + j\Lambda^T \Sigma_\Psi^{-1} \Lambda \mathbf{P}_0)^{-1} \mathbf{P}_0 \tag{4.222}$$

where **I** is the identity matrix. Using Schwartz's norm inequality one obtains

$$\|\mathbf{P}(j)\| \leq \|\mathbf{P}_0\| \cdot \|(\mathbf{I} + j\Lambda^T \Sigma_\Psi^{-1} \Lambda P_0)^{-1}\| \tag{4.223}$$

Then a lower bound M_1 on M is obtained from the inequality

$$\|(\mathbf{I} + j\mathbf{H}\Lambda^T \Sigma_\Psi^{-1} \Lambda \mathbf{P}_0)^{-1}\| \leq \varepsilon; \qquad j \geq M_1 \tag{4.224}$$

Obviously different matrix norms yield different bounds M_1 and exact results M.

Consider again the two link planar manipulator example. M may be easily determined from Equation 4.222 using PC-MATLAB. A few numerical results are shown below:

1. $r_1 = r_2 = 1$; $\theta_1(1) = 45°, \theta_2(1) = 33°$; $\theta_1(2) = 57°$; $\theta_2(2) = -5°$
   $\mathbf{P}_0 = \text{diag}(0.05, 0.05, 0.1, 0.1)$, $\varepsilon = 0.1$

$$\Sigma_\Psi = \text{diag}\left\{\begin{bmatrix} 0.2 & 0.1 \\ 0.1 & 0.2 \end{bmatrix}, \begin{bmatrix} 0.2 & 0.1 \\ 0.1 & 0.2 \end{bmatrix}\right\}$$

   Results: $M = 1377$, $M_1 = 1632$
   Both, P_0 and Σ_Ψ are "large."
2. Increasing the link lengths of the first example to $r_1 = 10$, $r_2 = 9$, keeping all the other values unchanged, results in $M = 995$.

3. Decreasing the measurement noise by a factor of 10:

$$\Sigma_{\Psi,3} = 0.1\Sigma_{\Psi,2} \tag{4.225}$$

Result: $M = 80$
So far the matrix norm that has been used corresponds to the 2-norm. Using the 1-norm (maximum column sum) gives $M = 100$. Using the infinity-norm (maximum row sum) gives $M = 86$.

4. Changing the robot configuration to

$$\theta_1(1) = 0°; \quad \theta_2(1) = 90°; \quad \theta_1(2) = 45°; \quad \theta_2(2) = -90°$$

gives $M = 2$ (for all three matrix norms).
Same result for the configuration:

$$\theta_1(1) = 0°; \quad \theta_2(1) = 0°; \quad \theta_1(2) = 45°; \quad \theta_2(2) = 180°$$

Obviously, there exists preferred configurations. In a qualitative sense, the more drastic the change in shape from one configuration to the next, the faster the convergence. The following group of examples studies configurations that differ only slightly.

5. The subscript of M denotes the matrix norm. It is also of interest to observe M_1 of the inequality in Equation 4.224. Two measurement covariance noise matrices have been tried, $\Sigma_{\Psi,3}$ and $0.1\Sigma_{\Psi,3}$. The robot configuration is described in terms of the 4-tuple $\theta_1(1)$, $\theta_2(1)$, $\theta_1(2)$, and $\theta_2(2)$ in degrees.

Configuration	Noise Covariance	$M_{(2)}$	$M_{1_{(2)}}$	$M_{(1)}$	$M_{(\infty)}$
(45, 20, 50, 10)	Medium	2748	2796	3328	2876
	Low	279	280	333	288
(46, 34, 58, −6)	Medium	62	62	77	67
	Low	7	7	8	7
(50, 34, 60, −15)	Medium	37	37	50	45
	Low	4	4	5	5
(0, 0, 180, 90)	Medium	3	3	4	4
	Low	1	1	1	1

In all situations, the spectral norm consistently gave the smallest M and the 1-norm the largest. The deviation between the exact M and the bound M_1 according to Equation 4.224 becomes significant only in cases of large numbers of repetitions.

As expected, low measurement noise combined with "suitable" measurement configuration drastically reduces the number of repetitions. This may go as low as $k_r = 1$.

4.6 IDENTIFICATION OF THE ROBOT JOINT AXES

The construction of a robot kinematic model, regardless of the specific modeling convention, always starts from the set of robot joint axes in an arbitrary robot configuration. While the nominal kinematic model of a robot may be obtained from engineering drawings of the machine or from crude measurements of the robot dimensions, finding a more exact kinematic model can be done through identification of the precise geometric relationships between the robot joint axes. For that, each joint of the robot is individually commanded to move to a goal point. During the motion of each joint, the position of a point on a tool mounted on the end effector (Figure 4.7) is determined with respect to a known reference

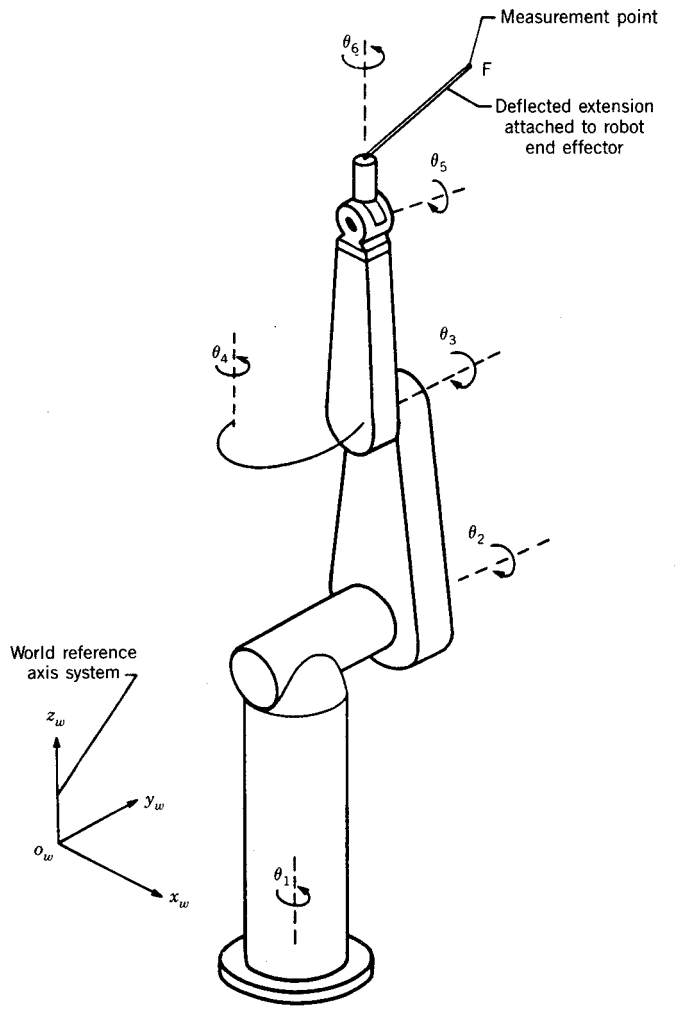


Figure 4.7. World axis system and robot arm with extension for measurement. Reprinted from [2].

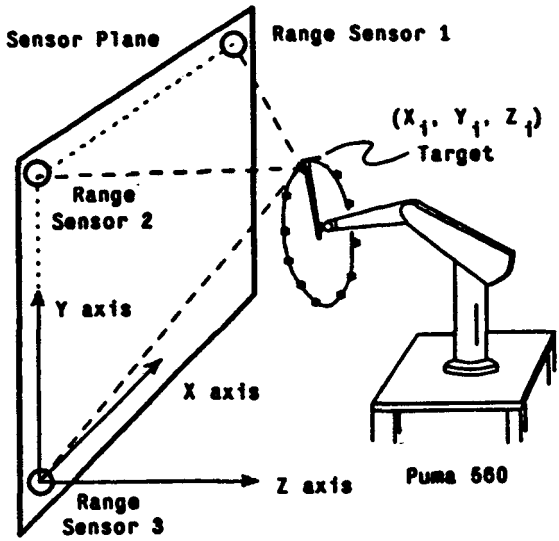


Figure 4.8. Generating a plane of rotation and center of rotation for joint 6 of a PUMA 560 robot. Reprinted with permission of the authors [27]. Copyright © 1987 IEEE.

frame. Any single sighting device can be used for that purpose. For a perfect revolute joint, the target point will move on a circular trajectory from which the corresponding axis of rotation can be estimated. The same method may be applied to prismatic joints.

The idea was developed independently by several researchers. Figure 4.7 shows a single-target robot set-up as was used by Barker [2]. Figure 4.8 shows the measurement set-up used at Carnegie Mellon University by Stone and Sanderson [27–29]. The target points in this set-up are transducers that generate an acoustic signal. These transducers are mounted on each robot link. The sensors are microphones that are placed at fixed locations in the world coordinate space. Each sensor measures the time required for the acoustic pulse to travel from the transducer to each sensor. This enables the target to be located by triangulation. This process is described in more detail in Chapter 3. The observed data points for each revolute joint motion define two geometric features. These are the *plane of rotation* and *center of rotation.* Regression methods are used to fit planes and circles through the observed target locations. The normal to the plane of rotation passing through the center of rotation is the joint axis. Similarly in the case of a prismatic joint, the locus of measured points define the *line-of-translation.* This line is parallel to the robot prismatic joint axis.

A similar method using a theodolite system was employed by researchers at the University of Texas at Austin [25, 26]. It was termed a *circle point analysis* (CPA). Whenever convenient, this short name will be used here as well.

The CPA method offers several potential advantages over the standard parameter identification strategy. First, no knowledge of the robot nominal model

is needed. The method can be used even in cases where kinematic parameter errors are relatively large.

Second, the identified joint axes may be expressed mathematically with respect to any convenient coordinate system since what matters is the relative position of every two consecutive lines. The lines are usually represented with respect to a coordinate frame defined by the sensory system.

The method can also be used to directly study joint imperfections such as "wobble" and "slop."

Finally, the method allows a comfortable use of a calibration tool mounted on or held by the robot hand where the repeatability in mounting the tool for calibration is not so critical.

One should also recognize some of the limitations of this approach. First, since only axes about which real motion occurs can be located in space, the method can be used to find the parameters between the first and last joint axes. The fixed transformations between the world coordinate frame and the robot base and between the end of the robot and the tool still need to be found by other means.

Second, the accuracy of the method increases the larger the range of joint motion during the data collection and the more uniformly spread apart the data points are. Three-dimensional measurement equipment that can cover only a small portion of the robot workspace often cannot be used to identify the robot axes of motion.

4.6.1 Linear Regression Analysis

Regression analysis deals with the relationship among measured variables. Such a relationship is expressed in terms of an equation relating one dependent variable y to one or more independent variables $x_1, x_2, \ldots, x_p$. If the number of independent variables is one the analysis is called "simple regression" compared to "multiple regression" in the case of several independent variables. Considering an equation that is linear in terms of the unknown coefficients

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p \tag{4.226}$$

the analysis is referred to as "linear regression analysis." The coefficients b_0, $b_1, \ldots, b_p$ are called the "regression coefficients."

This section provides formulas for the regression coefficients in both simple and multiple linear regression problems as a simple applications of least squares theory. The material is covered in many standard textbooks and references such as [5, 12].

Starting with simple linear regression, the measured data are a sequence of N pairs of values $(x_i, y_i), i = 1, 2, \ldots, N$. A typical situation is that of fitting a straight line to a set of measured points on a plane. The decision with regard to which variable should be taken as the dependent one is often arbitrary, and sometimes it depends on the physical nature of the problem.

The theoretical regression curve is:

$$y = \beta_0 + \beta_1 x \tag{4.227}$$

with β_0, β_1 being the unknown parameters.

Every data point (x_i, y_i) obeys the model

$$y_i = \beta_0 + \beta_1 x_i + u_i; \qquad i = 1, \ldots, N \tag{4.228}$$

where u_i is a noise term. The parameters β_0, β_1 are estimated using least squares to minimize the performance measure

$$J = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)^2 \tag{4.229}$$

The coefficients (b_0, b_1) are the optimal values of (β_0, β_1) that minimize J. In the context of least-square identification the aggregated measurement vector $\mathbf{Z}$ and the matrix coefficient $\mathbf{H}$ are

$$\mathbf{Z} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \tag{4.230}$$

and

$$\mathbf{H} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \tag{4.231}$$

One may now use the least-squares solution formula. It would, however, be simpler to differentiate J with respect to β_0 and β_1. The resulting solution is

$$b_1 = \frac{\sum_{i=1}^{N} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{4.232}$$

and

$$b_0 = \bar{y} - b_1 \bar{x} \tag{4.233}$$

where $\bar{y}$ and $\bar{x}$ are the mean values of y_i and x_i, respectively.

Note that the optimal straight line

$$y = b_0 + b_1 x \tag{4.234}$$

passes through the mean point $(\bar{x}, \bar{y})$.

Thus far, no assumptions have been made with regard to the noise term u_i in Equation 4.228. Under the common assumption that u_i, $i = 1, \ldots, N$ are Gaussian random variables that are independent of each other, have zero mean and a constant variance σ^2, a straightforward application of the variance formulas of least-squares estimation yield that b_0 and b_1 are unbiased estimates of β_0 and β_1 and that

$$\text{Var}(b_1) = \frac{\sigma^2}{\sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{4.235}$$

and

$$\text{Var}(b_0) = \sigma^2 \left[\frac{1}{N} + \frac{\bar{x}^2}{\sum_{i=1}^{N} (x_i - \bar{x})^2} \right] \tag{4.236}$$

Obviously, the larger the number of data points and the more spread apart they are, the better is the "goodness-of-fit" of the estimated line to the correct one.

Extensions of the above results to linear multiple regression problems is obtained as follows. N data points $(y_i, x_{1i}, \ldots, x_{pi})$ $i = 1, \ldots, N$ are measured. The theoretical regression curve is assumed to be

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \tag{4.237}$$

Every data point obeys the model

$$y_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi} + u_i \tag{4.238}$$

where u_i is the noise term. The regression parameters $\beta_0, \ldots, \beta_p$ are estimated via minimizing

$$H = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_{1i} - \cdots - \beta_p x_{pi})^2 \tag{4.239}$$

By differentiating J with respect to each one of the regression coefficients the following set of linear algebraic equations is obtained for the optimal regression parameters $b_1, \ldots, b_p$:

$$\mathbf{Sb} = \mathbf{a} \tag{4.240}$$

where

$$\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_p \end{bmatrix} \tag{4.241}$$

$\mathbf{S}$ is a $p \times p$ matrix whose entries are

$$S_{ij} = \sum_{k=1}^{N} (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j); \qquad (i,j) = 1, \ldots, p \tag{4.242}$$

The elements of the p-vector $\mathbf{a}$ are

$$a_i = \sum_{k=1}^{N} (y_k - \bar{y})(x_{ik} - \bar{x}_i); \qquad i = 1, \ldots, p \tag{4.243}$$

where

$$\bar{x}_i = \frac{1}{N} \sum_{k=1}^{N} x_{ik} \tag{4.244}$$

and

$$\bar{y} = \frac{1}{N} \sum_{k=1}^{N} y_k \tag{4.245}$$

The optimal coefficient b_0 is

$$b_0 = \bar{y} - b_1\bar{x}_1 - b_2\bar{x}_2 - \cdots - b_p\bar{x}_p \tag{4.246}$$

An alternative and equivalent solution is to use Equation 4.44 where

$$\mathbf{H} = \begin{bmatrix} 1 & x_{11} & x_{21} & \ldots & x_{p1} \\ 1 & x_{12} & x_{22} & \ldots & x_{p2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{1N} & x_{2N} & \ldots & x_{pN} \end{bmatrix} \tag{4.247}$$

and $\mathbf{Z}$ is as in Equation 4.230.

This formalism is useful for obtaining formulas for the variance of each one of the coefficients again under the same assumption that was made previously regarding the noise terms u_i.

4.6.2 Fitting Data Points to Planes, Circles, and Lines

This section focuses on curve fitting using linear regression analysis. More comprehensive treatment of the subject including additional techniques for curve fitting can be found in [28] and in its list of references.

4.6.2.1 Planes One form for the equation of a plane is

$$Ax + By + Cz + D = 0 \tag{4.248}$$

where the coefficients A, B, C, and D are the parameters to be identified. There are m data points (x_i, y_i, z_i), $i = 1, \ldots, m$. To use linear regression analysis any one of the variables x, y, or z may be selected as the dependent variable while the other two are considered as the independent variables.

Let z be chosen as the dependent variable. Equation 4.248 may now be rewritten as follows:

$$z = Ex + Fy + G \tag{4.249}$$

where E, F, and G are the unknown parameters that need to be chosen to minimize the following performance criterion:

$$J_z = \sum_{i=1}^{m} (z - z_i)^2 \tag{4.250}$$

In other words, the pairs of data points (x_i, y_i), $i = 1, \ldots, m$ are assumed to be the correct $X-Y$ coordinates of each point on the sought plane. Regression corresponds to a minimization of the sum of squared errors in the Z coordinate.

The formulas in the previous section are now specialized to plane estimation by observing the following variable changes in Equation 4.226 for the case $p = 2$

$$\begin{aligned} &y \leftrightarrow z, \quad \beta_0 \leftrightarrow G, \quad \beta_1 \leftrightarrow E, \quad \beta_2 \leftrightarrow F \\ &x_1 \leftrightarrow x, \quad x_2 \leftrightarrow y, \quad N \leftrightarrow m \end{aligned} \tag{4.251}$$

Then from Equations 4.240 through 4.246, the optimal parameters are

$$E = \frac{\sigma_{xx}\sigma_{yy} - \sigma_{zy}\sigma_{xy}}{\sigma_{xx}\sigma_{yy} - \sigma_{xy}^2} \tag{4.252}$$

$$F = \frac{\sigma_{xx}\sigma_{zy} - \sigma_{xy}\sigma_{zx}}{\sigma_{xx}\sigma_{yy} - \sigma_{xy}^2} \tag{4.253}$$

and

$$G = \bar{z} - E\bar{x} - F\bar{y} \tag{4.254}$$

where

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4.255}$$

$$\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i \tag{4.256}$$

$$\bar{z} = \frac{1}{m}\sum_{i=1}^{m} z_i \tag{4.257}$$

and:

$$\sigma_{xx} = \sum_{i=1}^{m} (x_i - \bar{x})^2 \tag{4.258}$$

$$\sigma_{yy} = \sum_{i=1}^{m} (y_i - \bar{y})^2 \tag{4.259}$$

$$\sigma_{xy} = \sum_{i=1}^{m} (x_i - \bar{x})(y_i - \bar{y}) \tag{4.260}$$

$$\sigma_{zx} = \sum_{i=1}^{m} (z_i - \bar{z})(x_i - \bar{x}) \tag{4.261}$$

$$\sigma_{zy} = \sum_{i=1}^{m} (z_i - \bar{z})(y_i - \bar{y}) \tag{4.262}$$

As explained earlier with regard to Equations 4.235 through 4.236, the larger the number of data points and the more spread apart the points are from each other, the better is the goodness-of-fit.

Finally, the closer the identified plane is to lying on the $X-Y$ plane of the coordinate system with respect to which the values (x_i, y_i, z_i) are given, the better the fit.

4.6.2.2 Circles on a Known Plane The analysis starts with the strong assumption that the data points (x_i, y_i, z_i), $i = 1, \ldots m$ all lie on a known (or previously identified) plane. Consequently, it is assumed that the coordinate frame, XYZ, with respect to which the data points are read is placed such that the plane lies on or parallel to the $X-Y$ plane defined by the coordinate frame. Thus one implicitly ignores z_i, or in other words only the (x_i, y_i) pairs are used in the identification.

The standard form for the equation of a circle is

$$(x - g)^2 + (y - h)^2 = r^2$$

where (g, h) are the X and Y coordinates of the circle center, respectively, and r is the radius.

To use regression analysis the dependent variable should depend linearly on the unknown coefficients.

Therefore, the circle equation is rewritten as follows:

$$w = x^2 + y^2 = Ax + By + C \tag{4.263}$$

where A, B, and C are the unknown regression coefficients. The selected dependent variable w is the squared distance between a point and the origin of the coordinate frame. The regression corresponds to minimizing the performance measure:

$$J_w = \sum_{i=1}^{m} (w - w_i)^2 \tag{4.264}$$

This minimization of the squared errors in w is in general not equivalent to minimizing the sum of squared perpendicular distances between the measurements and the circle unless the origin of the coordinate frame and the center of the circle coincide. The center location is unknown a priori and, therefore, the solution that minimizes distances to the circle circumference may be obtained through repeated solution of the problem $\min(J_w)$.

Using Equations 4.240 through 4.246 applied to the case $p = 2$, one uses the following changes of variables in Equation 4.226:

$$\begin{aligned} &y \leftrightarrow x^2 + y^2, \quad \beta_0 \leftrightarrow C, \quad \beta_1 \leftrightarrow A, \quad \beta_2 \leftrightarrow B \\ &x_1 \leftrightarrow x, \quad x_1 \leftrightarrow y, \quad N \leftrightarrow m \end{aligned} \tag{4.265}$$

The optimal solution is

$$A = \frac{\sigma_{wx}\sigma_{yy} - \sigma_{wy}\sigma_{xy}}{\sigma_{xx}\sigma_{yy} - \sigma_{xy}^2} \tag{4.266}$$

$$B = \frac{\sigma_{xx}\sigma_{wy} - \sigma_{xy}\sigma_{wx}}{\sigma_{xx}\sigma_{yy} - \sigma_{xy}^2} \tag{4.267}$$

$$C = \bar{w} - A\bar{x} - B\bar{y} \tag{4.268}$$

where $\bar{x}$, $\bar{y}$ are as in Equation 4.255 and 4.256, respectively. $\bar{w}$ is

$$\bar{w} = \frac{1}{m} \sum_{i=1}^{m} (x_i^2 + y_i^2) \tag{4.269}$$

σ_{xx}, σ_{yy}, and σ_{xy} are as in Equations 4.258 through 4.260

$$\sigma_{wx} = \sum_{i=1}^{m} (x_i^2 + y_i^2 - \bar{w})(x_i - \bar{x}) \tag{4.270}$$

$$\sigma_{wy} = \sum_{i=1}^{m} (x_i^2 + y_i^2 - \bar{w})(y_i - \bar{y}) \tag{4.271}$$

4.6.2.3 Lines in 3D The standard form for the equation of a line in three-dimensions is

$$\frac{x - x_0}{A} = \frac{y - y_0}{B} = \frac{z - z_0}{C} \tag{4.272}$$

where (x_0, y_0, z_0) are the coordinates of a point on the line. To characterize a line in space only four parameters are required. One seeks a line equation that minimizes the sum of the squared perpendicular distances between it and the measured points positions.

A linear regression solution strategy may involve the combination of two simple linear regression problems:

1. Fitting the projection line of Equation 4.272 onto the $X-Y$ plane using the measured (x_i, y_i) coordinates.
2. Fitting the projection line of Equation 4.272 onto the $X-Z$ plane using the measured (x_i, z_i) coordinates.

This is done in the following way. By Equations 4.232 through 4.233, the best straight line fit using (x_i, y_i), $i = 1, \ldots, m$ on the $X-Y$ plane is

$$y = b_0 + b_1 x \tag{4.273}$$

where

$$b_1 = \frac{\sigma_{xy}}{\sigma_{xx}} \tag{4.274}$$

and

$$b_0 = \bar{y} - b_1 \bar{x} \tag{4.275}$$

where σ_{xx}, σ_{xy} are as in Equations 4.258 and 4.260, respectively and $\bar{x}$, $\bar{y}$ are as in Equations 4.255 and 4.256, respectively.

Similarly, the best straight line fit using (x_i, z_i), $i = 1, \ldots, m$ on the $X-Z$ plane is

$$z = c_0 + c_1 x \tag{4.276}$$

where

$$c_1 = \frac{\sigma_{xz}}{\sigma_{xx}} \tag{4.277}$$

and

$$c_0 = \bar{z} - c_1 \bar{x} \tag{4.278}$$

where σ_{xz} and $\bar{z}$ are as in Equation 4.261 and 4.257, respectively.

Now, it is observed that Equation 4.273 is also the equation of a plane perpendicular to the $X-Y$ plane that contains the line given in Equation 4.273. Similarly, Equation 4.276 is the equation of a plane perpendicular to the $X-Z$ plane that contains the projection line in Equation 4.276. The intersection of the two planes defines the desired line (Equation 4.272) in 3D. In other words, the combined Equations 4.273 together with 4.276 constitute the 3D line.

Note that this line passes through the point $(\bar{x}, \bar{y}, \bar{z})$. To relate Equations 4.273 and 4.276 to Equation 4.272, the following parameters may be assigned.

$$x_0 = \bar{x}, \qquad y_0 = \bar{y}, \qquad z_0 = \bar{z} \tag{4.279}$$

$$A = \frac{1}{b_1}, \qquad B = 1, \qquad C = \frac{c_1}{b_1} \tag{4.280}$$

An alternative method is to use the principal axis method. For details, refer to work by Stone [28] and Ballard et al. [1].

4.6.3 Circle Point Analysis—The Measurement Phase

To find the axis of motion of joint j, a target point is placed on the $j+1$ link or on a tool attached rigidly to link $j+1$. That includes the possibility of placing the target on any of the links $j+2, j+3, \ldots$ under the condition that joints $j+1$, $j+2, \ldots$ do not move while joint j is moving. Thus, one may employ either a single target point located on the robot tool or end effector, or multiple target points located each on a different robot link. The target point that corresponds to the motion of joint j is denoted as the *jth target.*

The calibration measurement phase of an N degree of freedom manipulator may start at the Nth target analyzing the motion of the robot joint that is closest to the robot end effector. While measuring the position of the Nth target, joints 1 through $N-1$ are required to be in a fixed configuration.

After estimating the location of the Nth joint axis, the identification process focuses on the $N-1$ joint axis. For that, joints 1 through $N-2$ must remain at the same configuration as in the measurements of the Nth joint axis. Joint N, however, may be positioned arbitrarily. If target $N-1$ is also located at link $N+1$, then after selecting an appropriate position for joint N it must remain fixed during the motion of joint $N-1$.

This process is continued all the way down to joint 1. While measuring the position of the jth target, joints 1 through $j-1$ are required to be in their initial "arm signature" fixed configuration. Joints $j+1$ to N, on the other hand, can be positioned arbitrarily. Through independent control of the manipulator joints, joint j is then sequentially indexed to m different positions q_{ji}, $i = 1, \ldots, m$. To minimize the regression errors a good guideline is to keep these joint positions uniformly spaced about the entire range of motion of joint j. Then:

$$q_{ji} = q_{j,\min} + (i-1)\Delta q, \qquad i = 1, \ldots, m \tag{4.281}$$

where Δq is a constant joint increment and

$$q_{j,1} = q_{j,\min} \tag{4.282}$$

and

$$q_{j,m} = q_{j,\max} \tag{4.283}$$

The direction of motion from $q_{j,1}$ to $q_{j,m}$ corresponds to the positive sense of the corresponding joint motion, and so does the ordering of the measurement points. For each value of the joint position $q_{j,i}$ the coordinates (x_i, y_i, z_i) of the target are measured, $i = 1, \ldots, m$.

By following the above measurement procedure, the identification problem of each joint axis line equation becomes decoupled from the identification problems of the other joint axes. In the case where the target point j is not located on link $j + 1$, the configuration of joints $j + 1, \ldots, N$ may be taken in such a way to enhance the identification accuracy of axis j.

If, for instance, joint j is revolute, the configuration of the latter joints should be chosen to maximize the radius of the circle traversed by target j, or to maximize the circle segment that is visible to the sensory system.

Alternatively, the sequential robot joints motion may start at joint 1, "freezing" joints $2, \ldots, N$. The measurement then proceeds to joint 2, freezing joints $3, \ldots, N$ and choosing an appropriate fixed position for joint 1. So on all the way to joint N. This procedure was taken in [25] and is illustrated in Figure 4.9.

The entire identification procedure results in having the equations of N lines in space. These are the robot's N joint axes for the particular initial robot configuration characterized by a set of joint positions $q_1^s, q_2^s, \ldots, q_N^s$ (the "arm signature").

Depending on the particular sensory system that is used to trace the target points, an optimal selection of the arm signature may be done according to "line of visibility" considerations. This is the case when using opto-camera or three-cable techniques [16]. The identification accuracy depends on being able to trace the target along as large a portion of the total joint travel as possible. In the case of a revolute joint a practical guideline is to view the target along at least half of a circle.

The robot joint motion during the data collection experiment may be done continuously if all sensors are synchronized to read the target coordinates simultaneously. Otherwise there is a need to stop the robot at each point to allow each sensor to read the same point. Depending on the identification algorithm the joint position may or may not be read together with the target point coordinates. Synchronizing between the robot controller that reads the joint position sensors and the sensory system controller again may necessitate bringing the robot to a full stop at certain designated points along each joint travel.

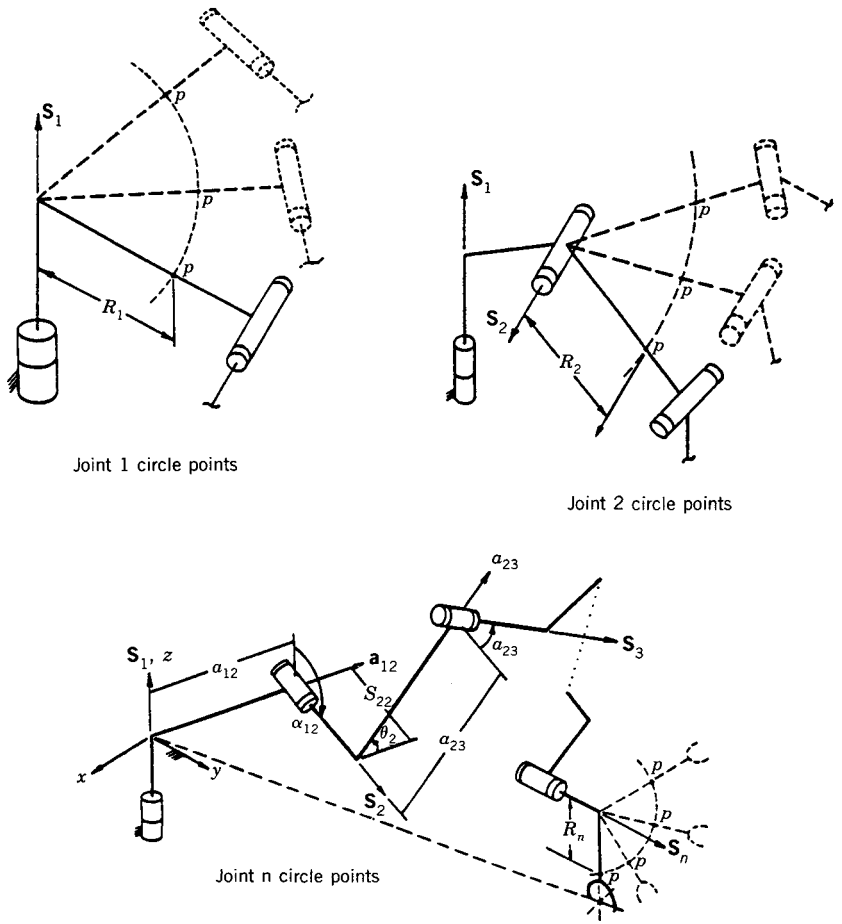


Figure 4.9. Circle point motion for individual axes, starting at joint 1 and ending with joint n. Adapted from reference [25] with permission of the author.

4.6.4 Iterative Identification of the Joint Axes

The material in this section follows closely Stone's analysis [28]. Focusing on the motion of one joint, let $\mathbf{p}_1, \ldots, \mathbf{p}_n$ be the coordinates of the target point with respect to the world coordinate frame established by the sensory system. Assuming a revolute joint, these n points ideally lie on a circle in three-dimensional space. Mathematically the problem of fitting a circle through the data point is a constrained nonlinear optimization problem as follows:

Minimize the function J:

$$J = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{p}_i\| \tag{4.284}$$

subject to the following constraints:

$$|\mathbf{x}_i - \mathbf{c}|^2 - r^2 = 0 \tag{4.285}$$

$$(\mathbf{x}_i - \mathbf{c})^T \mathbf{a} = 0 \tag{4.286}$$

and

$$|\mathbf{a}| - 1 = 0 \tag{4.287}$$

The vector $\mathbf{c}$ defines the center of the circle with respect to the world frame, the vector $\mathbf{a}$ defines unit vector normal to the plane in which the circle lies, r is the radius of the circle and the vectors $\mathbf{x}_i$, and $i = 1, \ldots, n$ are the positions of n points on the circle. The minimization is done with respect to $3n + 7$ parameters contained with $\mathbf{c}$, $\mathbf{a}$, r, $\mathbf{x}_1, \ldots, \mathbf{x}_n$. By using Lagrange multipliers the problem solution is obtained through solving $5n + 8$ nonlinear coupled algebraic equations. Since n is relatively large (between 50 and 200, in most practical applications) a direct identification solution is cumbersome.

To simplify the solution the problem is decomposed into a sequence of two subproblems:

1. Fitting the data points into a 3D plane to find the plane of rotation.
2. Fitting the data points into a circle that lies on the identified plane of rotation. By that the center of rotation is found.

As is shown in Section 4.6.2 each of the subproblems amounts to solving a linear regression problem. To ensure sufficient identification accuracy each one of the above subproblems needs to be solved iteratively as will be shown next.

Consider first the problem of fitting a plane through the data points. Taking z as the dependent variable the best goodness-of-fit is obtained if all the data points lie on the $X-Y$ plane or parallel to the $X-Y$ plane. Apriori, the plane of rotation of a particular joint under study in a given arm signature has no reason to be even close to lying parallel to the world $X-Y$ plane. The essence of the iterative solution is then to introduce a sequence of coordinate transformations to the frame with respect to which the data points are represented.

Let the revolute joint under study be j; $j = 1, \ldots, N$. Given the data points (x_i, y_i, z_i); $i = 1, \ldots, m$ of target j with respect to a world coordinate frame $X-Y-Z$, namely a coordinate frame that is determined by the measurement equipment or the sensors system, a coordinate transformation $\mathbf{T}$ from $X-Y-Z$ to $X'-Y'-Z'$ is initially needed to transform the measurement points to roughly lie either on the $X'-Y'$ plane or on a plane parallel to the $X'-Y'$ plane. The following strategy was attempted by Stone [28].

Referring to Figure 4.10 three of the measured positions that are mutually the most distant from one another are picked up. Denoting the points by $\mathbf{p}_k$, $\mathbf{p}_l$, and $\mathbf{p}_m$, where the corresponding joint positions satisfy $q_{j,k} < q_{j,l} < q_{j,m}$, a coordinate

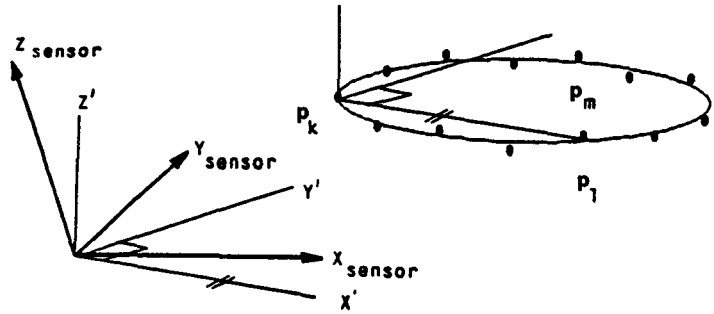


Figure 4.10. Initial approximation to the plane of rotation using three mutually distant target measurement. Reprinted with permission of Carnegie-Mellon University and the author [28].

frame $X'-Y'-Z'$, which shares the same origin with the frame $X-Y-Z$, is constructed as follows: The X' axis is taken to be parallel to the line joining $\mathbf{p}_k$ and $\mathbf{p}_l$. The Z' axis is taken to be perpendicular to this line and to the line joining $\mathbf{p}_k$ and $\mathbf{p}_m$. Thus, the transformation $\mathbf{T}$ becomes the following rotation transformation:

$$\mathbf{T} = \mathbf{R}_1 = \begin{bmatrix} \mathbf{n}_j & \mathbf{o}_j & \mathbf{a}_j & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.288}$$

where:

$$\mathbf{n}_j = \frac{\mathbf{p}_l - \mathbf{p}_k}{|\mathbf{p}_l - \mathbf{p}_k|} \tag{4.289}$$

$$\mathbf{a}_j = \frac{(\mathbf{p}_l - \mathbf{p}_k) \times (\mathbf{p}_m - \mathbf{p}_k)}{|(\mathbf{p}_l - \mathbf{p}_k) \times (\mathbf{p}_m - \mathbf{p}_k)|} \tag{4.290}$$

and

$$\mathbf{o}_j = \mathbf{a}_j \times \mathbf{n}_j \tag{4.291}$$

Denoting the transformed measurement points by $\mathbf{p}^1_{j,i}$, $i = 1, \ldots, m$. These are obtained from the original data points $\mathbf{p}_{j,i}$ as follows:

$$\mathbf{p}^1_{j,i} = \mathbf{R}_1 \mathbf{p}_{j,i}, \qquad i = 1, \ldots, m \tag{4.292}$$

$\mathbf{p}_{j,i}$ or $\mathbf{p}^1_{j,i}$ are represented as 4-vectors to conform to the homogeneous transformation formulation given in Equation 4.288.

The second step after transforming the measurement points to the $X-Y-Z$ frame is to fit the points to a plane using Equations 4.248 and 4.252 through

4.262. The result of the regression analysis is a plane characterized by the coefficients A^0, B^0, C^0, D^0 with respect to the $X'-Y'-Z'$ frame, namely $A^0x' + B^0y' + C^0z' + D^0 = 0$.

The next step is the representation of the plane with respect to the $X-Y-Z$ frame. Since $\mathbf{R}_1$ is a pure rotation, this is done in the following way:

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \mathbf{R}_1^T \begin{bmatrix} A^0 \\ B^0 \\ C^0 \\ D^0 \end{bmatrix} \tag{4.293}$$

As a result of the first iteration the estimated plane of rotation may still not be parallel to the $X'-Y'$ plane. A new coordinate frame $X''-Y''-Z''$ is now defined by taking a rotation transformation R_2 with respect to the $X-Y-Z$ frame. R_2 is constructed using the most current estimate of the plane of rotation. A plane $Ax + By + Cz + D = 0$ defines a vector $(A, B, C, 1)^T$ that is normal to that plane. Thus let the Z'' axis unit direction vector $\mathbf{a}_j$ be given by the estimated normal to the plane of rotation:

$$\mathbf{a}_j = \begin{bmatrix} \frac{A}{w} \\ \frac{B}{w} \\ \frac{C}{w} \end{bmatrix} \tag{4.294}$$

where

$$w = (A^2 + B^2 + C^2)^{1/2} \tag{4.295}$$

The X'' and Y'' axes unit direction vectors $\mathbf{n}_j$ and $\mathbf{o}_j$, respectively, are chosen arbitrarily to form a right-hand system.

The original set of points $\mathbf{p}_{j,i}$ is now transformed using $\mathbf{R}_2$, and a new plane A^1, B^1, C^1, D^1 with respect to $X''-Y''-Z''$ is found using regression analysis.

The process is used repeatedly until the difference between consecutive estimates of the plane of rotation becomes negligible. Stone reports that under the following practical assumptions only a few iterations are required for the algorithm to converge.

Assumption 1: The standard deviaton in the measurements of the target's cartesian position is by several orders of magnitude smaller than the nominal radius of rotation of the target.

Assumption 2: The measured target positions correspond to joint positions uniformly distributed between the upper and lower limits of the joints travel.

Assumption 3: For each revolute joint, at least 180° of joint travel is traversed as part of the data collection.

The next step after estimating the plane of rotation is to estimate the center of rotation. Let K denote the number of the last iteration in estimating the plane of rotation. Then $\mathbf{p}_{j,i}^{K}$, $i = 1, \ldots, m$ are the measurement point representations with respect to the coordinate frame $X^{(K)} - Y^{(K)} - Z^{(K)}$ frame. This circle does not minimize the sum of squared perpendicular distances between the measurements and the circle as one would expect. A sequence of iterative solutions is now initiated involving a sequence of translation transformations $X^{(K)} - Y^{(K)} - Z^{(K)} \rightarrow X^{(K+1)} - Y^{(K+1)} - Z^{(K+1)} \cdots$ in such a way that the origin of the new coordinate frame coincides with the most current estimate of the circle center. This is done as follows:

Let the circle equation after iteration k be

$$(x - g_k)^2 + (y - h_k)^2 = r_k^2 \tag{4.296}$$

where x, y are represented in the $X^{(k)} - Y^{(k)} - Z^{(k)}$ coordinate frame. $k = K$, $K + 1, \ldots, K + L$. Each one of the data points $\mathbf{p}_1^{(k)}, \ldots, \mathbf{p}_m^{(k)}$ represented with respect to the kth coordinate frame is translated by $(g_k, h_k, 0)^T$. In other words frame $k + 1$ is obtained from frame k via

$$\mathbf{F}_{k+1} = \begin{bmatrix} 1 & 0 & 0 & g_k \\ 0 & 1 & 0 & h_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{F}_k \tag{4.297}$$

The algorithm terminates when the changes in the location of the circle center become very small.

Finally, the circle equation parameters $g^{(K+L)}$, $h^{(K+L)}$, $r^{(K+L)}$ should be transformed back to the world frame X–Y–Z retracting the last L translative transformations that have been made previously.

The joint axes line equations can now be written in terms of a point on the line, this is the center of rotation coordinates in the case of a revolute joint denoted as (x_{cj}, y_{cj}, z_{cj}) and a unit vector normal to the plane of rotation given in terms of the element of $\mathbf{a}_j$ of Equation 4.294. Thus

$$\frac{x - x_{cj}}{A_j/w_j} = \frac{y - y_{cj}}{B_j/w_j} = \frac{z - z_{cj}}{C_j/w_j}; \qquad j = 1, \ldots, N \tag{4.298}$$

The estimation of a prismatic joint axis may follow similar ideas. The "best-fit" line must pass through the mean point $(\bar{x}, \bar{y}, \bar{z})$. Thus, the data points may all be initially translated to a new frame having $(\bar{x}, \bar{y}, \bar{z})$ as its origin.

4.6.5 Kinematic Parameter Extraction from Identified Joint Axes

The circle point analysis data collection and identification actions result in obtaining least-squares estimates for the line equations of all robot joint axes at a given robot configuration termed the arm signature. In the case of a revolute joint the identified center of rotation provides a point on the identified joint axis, whereas the identified plane of rotation provides for the direction cosines of the joint axis line. In the case of a prismatic joint the mean point is on the estimated joint axis line. The identified line is the intersection of two identified perpendicular planes.

The main objective of the circle point analysis method is the identification of the robot kinematic parameters. The selection of a suitable arm signature plays an important role in determining the regression goodness-of-fit of the joint axes line equations. Once these lines are obtained, the extraction of the robot link parameters becomes independent of the choice of the arm signature.

Two strategies are offered in the literature to extract the kinematic parameters from the identified joint axes. The first, due to Stone [28], consists of the following steps:

1. Link coordinate frame specification
2. Calculation of the link homogeneous transformation matrices from the relative locations of the assigned link coordinate frames
3. Derivation of analytic formulas for the kinematic parameters in terms of elements of the link homogeneous transformation matrices. The formulas are derived in a method very similar to Paul's [23] inverse kinematics solution method.

The second strategy due to Sklar [25, 26] is to compute geometric entities such as common normal lengths, offset distances, and twist angles directly from the identified joint axes line equations. Standard vector algebra relationships and solid geometry formulas are used for this purpose.

4.6.5.1 Stone's Method for Kinematic Parameter Extraction from Identified Joint Axes Given an identified joint axis line equation expressed in terms of a world coordinate frame normally defined by the calibration sensory system, a three-step process is proposed for extracting the kinematic parameters of interest.

The first step involves specification of link coordinate frames. The modeling convention known also as the S-Model (refer to Chapter 2) consists of

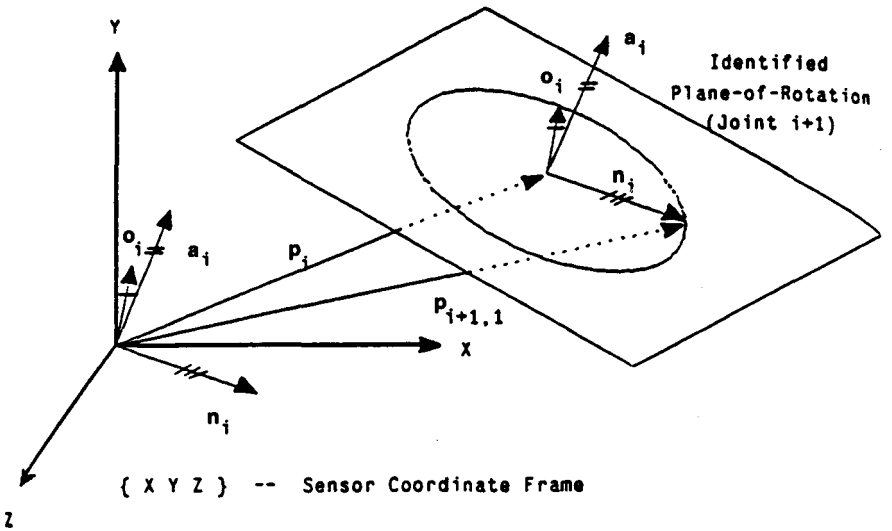


Figure 4.11. Coordinate frame construction for a revolute joint. Reprinted with permission of Carnegie-Mellon University and the author [28].

1. Assigning the Z_i axis for link i frame to be along the identified joint axis $i + 1$.
2. Assigning the origin of frame i to be at the identified center of rotation of axis $i + 1$ (in the case of a revolute joint) or at the mean point (in the case of a prismatic joint).
3. Arbitrary assignment of the X_i, Y_i axes over the plane perpendicular to joint axis $i + 1$. In the case of a revolute joint this is the identified plane of rotation.

Figures 4.11 and 4.12 illustrate the link coordinate frame assignment in the cases of a revolute joint and a prismatic joint, respectively. In the case of a revolute joint, the unit direction vector $\mathbf{a}_i$ is the unit normal vector to the estimated plane of rotation of joint $i + 1$. Its positive sense must correspond with the positive sense of rotation of the $i + 1$ joint. The direction of either the X axis, defined by $\mathbf{n}_i$, or the Y axis, defined by $\mathbf{o}_i$ is arbitrary in the S-model. For instance, one may choose the unit direction vector $\mathbf{n}_i$ to be

$$\mathbf{n}_i = \frac{\mathbf{p}_{i+1,1} - \mathbf{p}_{i+1,c}}{|\mathbf{p}_{i+1,1} - \mathbf{p}_{i+1,c}|} \tag{4.299}$$

where $\mathbf{p}_{i+1,1}$ is the location of the first target position for joint $i + 1$ projected onto the plane of rotation. $\mathbf{p}_{i+1,c}$ is the center of rotation. The Y axis direction vector $\mathbf{o}_i$ is the cross-product of the Z and X axis unit direction vectors, $\mathbf{a}_i$ and

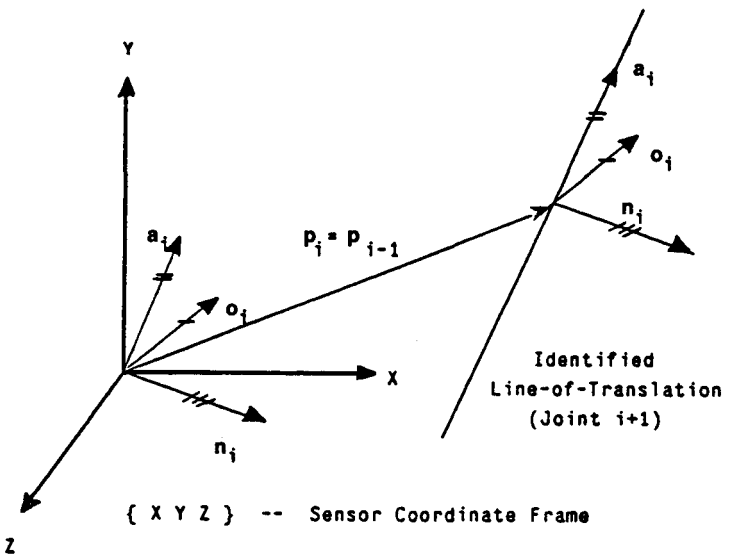


Figure 4.12. Coordinate frame construction for a prismatic joint. Reprinted with permission of Carnegie-Mellon University and the author [28].

$\mathbf{n}_i$, respectively. The unit direction vector $\mathbf{a}_i$, in the case of a prismatic joint, is the unit direction vector of the estimated line of translation of the $i + 1$ joint. Again, the positive sense of $\mathbf{a}_i$ conforms to the positive sense of the joint translation. The choice of either the X axis or the Y axis is completely arbitrary.

The second step, after assigning link coordinate frames, involves the computation of the link homogeneous transformation matrices. Let $\mathbf{S}_i$ denote the homogeneous transformation relating coordinate frame i to the world coordinate frame. Then

$$\mathbf{S}_i = \begin{bmatrix} \mathbf{n}_i & \mathbf{o}_i & \mathbf{a}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.300}$$

Denoting by $\mathbf{B}_i$, $i = 1, \ldots, N$ the relative link homogeneous transformation and by $\mathbf{B}_0$ the fixed transformation relating the robot base to the world coordinate frame, we have:

$$\mathbf{S}_i = \mathbf{B}_0\mathbf{B}_1 \ldots \mathbf{B}_i; \qquad i = 1, \ldots, N \tag{4.301}$$

Thus

$$\mathbf{B}_i = \mathbf{S}_{i-1}^{-1}\mathbf{S}_i; \qquad i = 2, \ldots, N \tag{4.302}$$

which can be computed without the knowledge of $\mathbf{B}_0$. The computation of $\mathbf{B}_1$ requires having $\mathbf{B}_0$, and then

$$\mathbf{B}_1 = \mathbf{B}_0^{-1}\mathbf{S}_1 \tag{4.303}$$

The robot calibration process involves the estimation of $\mathbf{B}_0$ as well. This must be established before the circle point analysis starts.

The $\mathbf{B}_i$ transformations are analogous to the Denavit–Hartenberg $\mathbf{A}_i$ transformation matrices (see Chapter 2). However, because the assignment of the origin of the link coordinate frame and its X axis are arbitrary, there are two additional link parameters.

$$\mathbf{B}_i = \mathbf{R}(z, \beta_i)\mathbf{T}(0, 0, \bar{r}_i)\mathbf{T}(l_i, 0, 0)\mathbf{R}(x, \alpha_i)\mathbf{R}(z, y_i) \tag{4.304}$$

where l_i is the common normal between axes i and $i + 1$ and α_i is the twist angle between axes i and $i + 1$. Also

$$\beta_i = \theta_i - \gamma_{i-1} \tag{4.305}$$

and

$$\bar{r}_i = r_i - b_{i-1} \tag{4.306}$$

where θ_i is the rotation angle between X_{i-1} and X_i and r_i is the offset distance between l_{i-1} and l_i. The additional parameters γ_i and b_i account for the differences in placing the link coordinate frame as compared to the DH convention.

Expansion of Equation 4.304 results in the following expressions:

$$\mathbf{B}_i = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_x \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.307}$$

where

$$n_x = \cos\beta_i \cos\gamma_i - \sin\beta_i \cos\alpha_i \sin\gamma_i \tag{4.308}$$

$$n_y = \sin\beta_i \cos\gamma_i + \cos\beta_i \cos\alpha_i \sin\gamma_i \tag{4.309}$$

$$n_z = \sin\alpha_i \sin\gamma_i \tag{4.310}$$

$$o_x = -\cos\beta_i \sin\gamma_i - \sin\beta_i \cos\alpha_i \cos\gamma_i \tag{4.311}$$

$$o_y = -\sin\beta_i \sin\gamma_i + \cos\beta_i \cos\alpha_i \cos\gamma_i \tag{4.312}$$

$$o_z = \sin\alpha_i \cos\gamma_i \tag{4.313}$$

$$a_x = \sin\beta_i \sin\alpha_i \tag{4.314}$$

$$a_y = -\cos\beta_i \sin\alpha_i \tag{4.315}$$

$$a_z = \cos\alpha_i \tag{4.316}$$

$$p_x = b_i \sin\beta_i \sin\alpha_i + a_i \cos\beta_i \tag{4.317}$$

$$p_y = -b_i \cos\beta_i \sin\alpha_i + a_i \sin\beta_i \tag{4.318}$$

$$p_z = b_i \cos\alpha_i + d_i \tag{4.319}$$

Given the numerical values of the $\mathbf{B}_i$ matrices, the last step involves solving Equations 4.308 through 4.319 for the six unknown link parameters.

Applying Paul's backward multiplication technique [23] the solutions are

$$\beta_i = \begin{cases} 0 & \text{when } a_x = a_y = 0 \\ \operatorname{atan}\dfrac{-a_x}{a_y} & \text{otherwise} \end{cases} \tag{4.320}$$

$$\alpha_i = \operatorname{atan}\frac{a_x^2 + a_y^2}{a_z} \tag{4.321}$$

$$\gamma_i = \operatorname{atan}\frac{(a_x n_z)^2 + (a_y n_z)^2 + (a_y n_y + a_x n_x)^2}{-a_y n_x + a_x n_y} \tag{4.322}$$

$$a_i = p_x \cos\beta_i + p_y \sin\beta_i \tag{4.323}$$

$$b_i = \begin{cases} \dfrac{-p_x \sin\beta_i + p_y \cos\beta_i}{\sin\alpha_i} & \text{if } \sin\alpha_i \neq 0 \\ 0 & \text{if } \sin\alpha_i = 0 \end{cases} \tag{4.324}$$

$$d_i = p_z - b_i \cos\alpha_i \tag{4.325}$$

from which the DH parameters may be extracted.

4.6.5.2 Sklar's Method for Kinematic Parameter Extraction from Identified Joint Axes Following closely the analysis done by Sklar [25], this section focuses on extracting the DH parameters directly from the joint axes line equations. The ideas can be easily extended to include kinematic parameters used in other robot kinematic modeling conventions.

The following notation is used in this section:

$\mathbf{a}_{ij}$ = direction vector of the common normal between joint axes i and j, pointing from axis i to axis j. In the case where $j = i + 1$, a shorthand notation $\mathbf{a}_i$ will be used.

$\mathbf{s}_i$ = direction vector pointing along the positive direction of joint axis i.

The available data for the kinematic parameter extraction include all direction vectors $\mathbf{s}_i$, $i = 1, \ldots, N$, and a given point (x_i, y_i, z_i) lying on each joint axis i.

We shall denote such points by the vector $\mathbf{r}_i$.

$$\mathbf{r}_i = (x_i, y_i, z_i)^T \tag{4.326}$$

Denote by α_{ij} the twist angle between axes i and j, as measured by a right-handed rotation from $\mathbf{s}_i$ to $\mathbf{s}_j$ about the vector $\mathbf{a}_{ij}$. The angle α_{ij} can be computed from the pair of equations:

$$\cos \alpha_{ij} = \mathbf{s}_i \cdot \mathbf{s}_j \tag{4.327}$$

$$\sin \alpha_{ij} = (\mathbf{s}_i \times \mathbf{s}_j) \cdot \mathbf{a}_{ij} \tag{4.328}$$

In other words, the magnitudes of the twist angles, α_i, are readily available from the given joint axis descriptions. Their signs, however, require finding first the appropriate common normal vectors.

The common normal vectors are also needed for computing the joint rotation angle θ_j. This angle is measured by a right-hand rotation from $\mathbf{a}_{ij}$ to $\mathbf{a}_{jk}$ about the vector $\mathbf{s}_j$, as follows:

$$\cos \theta_j = \mathbf{a}_{ij} \cdot \mathbf{a}_{jk} \tag{4.329}$$

$$\sin \theta_j = (\mathbf{a}_{ij} \times \mathbf{a}_{jk}) \cdot \mathbf{s}_j \tag{4.330}$$

In the case of a revolute joint j, the calculated angle θ_j corresponds to the particular arm signature joint j position.

The remainder of this section focuses therefore on the computation of the common normals between each pair of consecutive robot joint axes and the offset distance between consecutive common normal vectors.

When studying joint axes i and $i + 1$, three cases are distinguished and different formulas are derived for each case. The cases are

1. Skew lines
2. Intersecting lines
3. Parallel lines.

The appropriate case can be determined from the given experimental data. Such a test is developed next.

A line in space, as shown in Figure 4.13 has the following equation written in terms of its direction vector $\mathbf{s}_i$, a fixed point $\mathbf{r}_i$ and an arbitrary point $\mathbf{r}$:

$$(\mathbf{r} - \mathbf{r}_i) \times \mathbf{s}_i = 0 \tag{4.331}$$

The constant vector $\mathbf{s}_{0i}$ defined as

$$\mathbf{s}_{0i} = \mathbf{r}_i \times \mathbf{s}_i \tag{4.332}$$

represents the moments of each component of the direction vector $\mathbf{s}_i$ about the coordinate axes.

The basic equation used to indicate if a pair of lines are skew, parallel, or intersecting is referred to as the *mutual moment* (*MM*) equation. The mutual

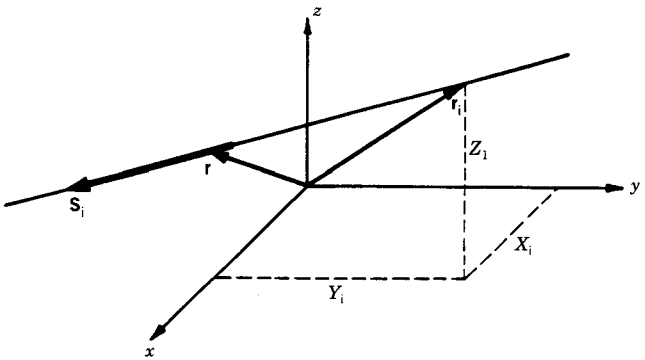


Figure 4.13. A general line in space. Adapted from reference [25] with permission of the author.

moment of two lines j and k is defined as

$$MM = \mathbf{s}_j \cdot \mathbf{s}_{0k} + \mathbf{s}_k \cdot \mathbf{s}_{0j} \tag{4.333}$$

MM is the moment of $\mathbf{s}_k$ on $\mathbf{s}_j$, or symmetrically the moment of $\mathbf{s}_j$ on $\mathbf{s}_k$. Referring to Figure 4.14, let $\mathbf{P}_k$ and $\mathbf{P}_j$ be the intersection points of the common normal between line k and line j with the respective lines. Equation 4.333 results from expanding the expression

$$MM = [(\mathbf{P}_k - \mathbf{P}_j) \times \mathbf{s}_k] \cdot \mathbf{s}_j. \tag{4.334}$$

For details refer to the work by Sklar [25]. Obviously

$$\mathbf{P}_k - \mathbf{P}_j = a_{jk}\mathbf{a}_{jk} \tag{4.335}$$

where a_{jk} is the common normal length. By substituting Equation 4.335 into Equation 4.334, and using the twist angle Equation 4.328, the mutual moment equation is finally obtained:

$$MM = -a_{jk} \sin \alpha_{jk} \tag{4.336}$$

where MM is computed from the given measured data using Equations 4.332 and 4.333. Equation 4.336 provides the conditions to distinguish among the three cases of an arbitrary pair of lines. The lines are skew if $MM \neq 0$. When $MM = 0$ the lines are either intersecting (if $a_{jk} = 0$) or parallel (if $\alpha_{jk} = 0$). Distinguishing between the intersecting and parallel cases can be done through the twist angle Equation 4.327.

A second application of Equation 4.336 is in determining the sign of the twist angle. If $MM > 0$ then $\alpha_{jk} > 180°$, and if $MM < 0$ then $a_{jk} < 180°$. Again refer to Figure 4.14.

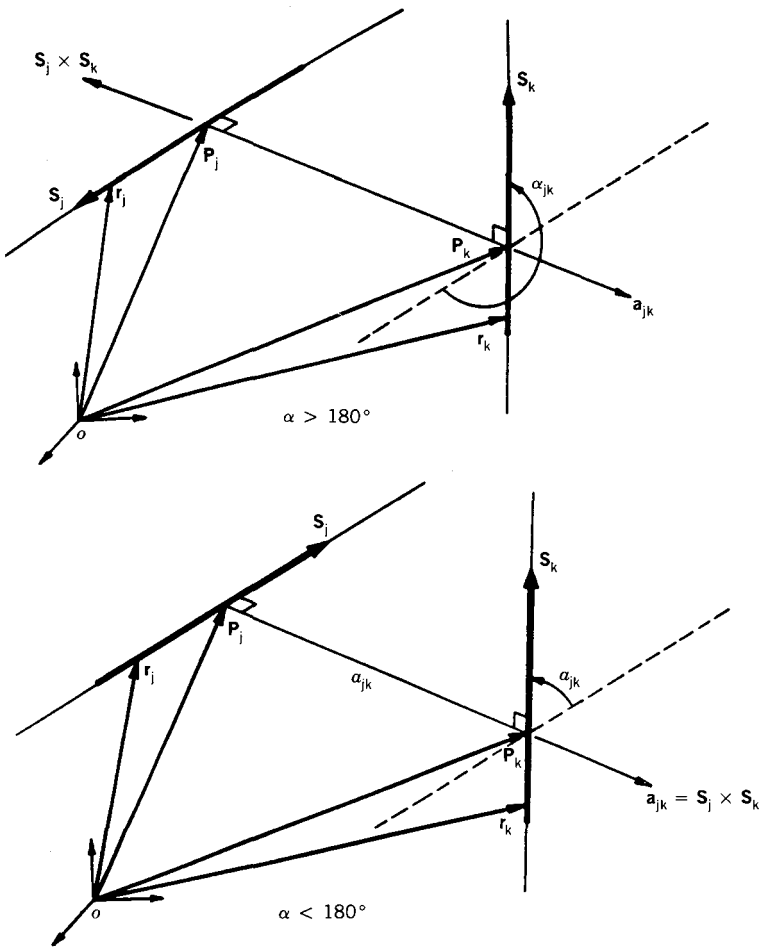


Figure 4.14. General cases of adjacent lines. Adapted from reference [25] with permission of the author.

A third application of the mutual moment equation is in determining the direction sign of the common normal vector, whenever $MM \neq 0$.

$$\mathbf{a}_{jk} = -\text{sign}(MM)\frac{\mathbf{s}_j \times \mathbf{s}_k}{|\mathbf{s}_j \times \mathbf{s}_k|} \tag{4.337}$$

The joint rotation angles θ_j can now be found using Equations 4.329 and 4.330.

The common normal length a_{jk} can be calculated projecting an arbitrary vector defined in terms of two points, one that lies on axis j and the other that lies on axis k, onto the common normal unit direction vector. Let $\mathbf{A}_j$ be an arbitrary given point on axis j, and $\mathbf{A}_k$ be an arbitrary point on axis k, then

$$a_{jk} = (\mathbf{A}_j - \mathbf{A}_k) \cdot \mathbf{a}_{jk} \tag{4.338}$$

To find the offset distance d_j between every two consecutive common normal vectors $\mathbf{a}_{ij}$ and $\mathbf{a}_{jk}$, it is necessary to compute the coordinates of the points at which each common normal intersects the joint axes, namely points $\mathbf{P}_j$ and $\mathbf{P}_k$ (as denoted in Figure 4.14). Two equivalent computation methods for finding these intersection points are shown next.

Method 1 (Sklar [25]): Considering first the case of skew lines, the intersection point $\mathbf{P}_j$ between the common normal $\mathbf{a}_{jk}$ and axis j, is also the intersection point between axis j and a plane defined by $\mathbf{a}_{jk}$ and $\mathbf{s}_k$. Solid geometry theory provides for a general formula for the intersection point between a given plane and a given line, as follows:

$$\mathbf{P}_j = \frac{(\mathbf{a}_{jk} \times \mathbf{s}_k) \times \mathbf{s}_{0j} - (\mathbf{s}_{0k} \cdot \mathbf{a}_{jk})\mathbf{s}_j}{(\mathbf{a}_{jk} \times \mathbf{s}_k) \cdot \mathbf{s}_j} \tag{4.339}$$

Similarly

$$\mathbf{P}_k = \frac{(\mathbf{a}_{jk} \times \mathbf{s}_k) \times \mathbf{s}_{0k} - (\mathbf{s}_{0j} \cdot \mathbf{a}_{jk})\mathbf{s}_j}{(\mathbf{a}_{jk} \times \mathbf{s}_j) \cdot \mathbf{s}_k} \tag{4.340}$$

Denoting by $\mathbf{Q}_i$ and $\mathbf{Q}_j$ the intersection points of common normal $\mathbf{a}_{ij}$ with axes i and j, respectively, the offset distance r_j is obtained from

$$r_j = (\mathbf{P}_j - \mathbf{Q}_j) \cdot \mathbf{s}_j \tag{4.341}$$

For intersecting axes, we have

$$\mathbf{P}_k = \mathbf{P}_j \tag{4.342}$$

Taking the cross-product of both sides of Equation 4.342 with the vector $\mathbf{s}_k$ yields

$$\mathbf{P}_j \times \mathbf{s}_k = \mathbf{s}_{0k} \tag{4.343}$$

Taking now the cross-product of both sides of Equation 4.343 with the vector $\mathbf{s}_{0j}$ isolates the unknown $\mathbf{P}_j$. The result after several simplification steps becomes

$$\mathbf{P} = \mathbf{P}_k = \mathbf{P}_j = \frac{\mathbf{s}_{0j} \times \mathbf{s}_{0k}}{\mathbf{s}_{0j} \cdot \mathbf{s}_k} = \frac{\mathbf{s}_{0k} \times \mathbf{s}_{0j}}{\mathbf{s}_j \cdot \mathbf{s}_{0k}} \tag{4.344}$$

In the case of parallel axes j and k, $r_j = 0$ by definition of the DH convention. One still needs to determine a_{ij} and $\mathbf{a}_{ij}$ for this case. This is done as follows:

$$\mathbf{P}_j + a_{jk}\mathbf{a}_{jk} = \mathbf{P}_k \tag{4.345}$$

Taking the cross-product with the vector $\mathbf{s}_j$ yields

$$\mathbf{s}_{0j} + a_{jk}\mathbf{a}_{jk} \times \mathbf{s}_j = \cos\alpha_{jk}\mathbf{s}_{0k} \tag{4.346}$$

where $\cos\alpha_{jk} = \pm 1$. The exact sign of $\cos\alpha_{jk}$ is known through a previously made computation. Another cross-product with $\mathbf{s}_j$ using the vector algebra relationship

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a}\cdot\mathbf{c})\mathbf{b} - (\mathbf{a}\cdot\mathbf{b})\mathbf{c} \tag{4.347}$$

isolates the common normal vector as follows:

$$a_{jk}\mathbf{a}_{jk} = \mathbf{s}_j \times (\cos\alpha_{jk}\mathbf{s}_{0k} - \mathbf{s}_{0j}) \tag{4.348}$$

Method 2: The starting point of the alternative method for computing the intersection points $\mathbf{P}_j$, $\mathbf{P}_k$ between the $j-k$ common normal and the j, k axes is a parametric representation of the joint axes lines. Let $\mathbf{P}$ be an arbitrary point on axis j, and $\mathbf{P}_{cj}$ be a known point on that line.

Then

$$\mathbf{P} = \mathbf{P}_{cj} + \mathbf{s}_j t \tag{4.349}$$

where t is a real parameter. Let $t = t_j$ denote the parameter value at the intersection point, $\mathbf{P}_j$.

$$\mathbf{P}_j = \mathbf{P}_{cj} + \mathbf{s}_j t_j \tag{4.350}$$

Similarly, let v be the free parameter in describing axis k, where $v = 0$ corresponds to a known point $\mathbf{P}_{c,k}$, and $v = v_k$ corresponds to the unknown intersection point $\mathbf{P}_k$.

By orthogonality of the pairs ($\mathbf{a}_{jk}$ and $\mathbf{s}_j$) and ($\mathbf{a}_{jk}$ and $\mathbf{s}_k$), two *linear* algebraic equations in terms of the unknown variables t_j and v_k are obtained as follows:

$$(\mathbf{P}_k - \mathbf{P}_j)\cdot\mathbf{s}_j = (\mathbf{P}_k - \mathbf{P}_j)\cdot\mathbf{s}_k = 0 \tag{4.351}$$

Substitution of Equations 4.349 and 4.350 into Equation 4.351 yields

$$(L_j^2 + M_j^2 + N_j^2)t_j - (L_jL_k + M_jM_k + N_jN_k)v_k = (x_{c,k} - x_{c,j})L_j + (y_{c,k} - y_{c,j})M_j + (z_{c,k} - z_{c,j})N_j \tag{4.352}$$

$$(L_jL_k + M_jM_k + N_jN_k)t_j - (L_K^2 + M_k^2 + N_k^2)v_k = (x_{c,k} - x_{c,j})L_k + (y_{c,k} - y_{c,j})M_k + (z_{c,k} - z_{c,j})N_k \tag{4.353}$$

where

$$\mathbf{s}_j = (L_j, M_j, N_j)^T \tag{4.354}$$

$$\mathbf{s}_k = (L_k, M_k, N_k)^T \tag{4.355}$$

and $(x_{c,j}, y_{c,j}, a_{c,j})^T$ and $(x_{c,k}, y_{c,k}, z_{c,k})^T$ are the known points on axes j and k, respectively.

4.7 OBSERVABILITY ISSUES IN KINEMATIC ERROR PARAMETER IDENTIFICATION OF BASE AND TOOL TRANSFORMATIONS

4.7.1 Introduction

Better understanding of the identifiability of parameters in the base and tool transformations requires a somewhat more careful approach to robot error model construction. It is shown in this section that a consistent differential transformation formalism needs to be taken to construct error models in order to ensure that the identification Jacobian is a matrix function of *all* joint variables and that the joint variables are about the *actual* joint axes of the manipulator. It is shown that if robot calibration measurements do not provide the information about the end effector's orientation, the orientation parameters in either the base or the tool transformations become unobservable. It is also shown that under additional conditions of the tool or world frames, a further reduction in the number of observable kinematic parameters may occur, depending on the choice of error models. The section also investigates the relationship between common observability measures and the irreducibility of the error model.

This section is organized as follows: 4.7.2 focuses on some properties of left and right differential transformations. These are used in Section 4.7.3 to derive two versions of generic manipulator kinematic error models. These models are "generic" in the sense of being independent of particular choices of manipulator kinematic modeling conventions. In Section 4.7.4, the concept of error model irreducibility is introduced and discussed. The generic error models are utilized in Section 4.7.5 to study several issues in the kinematic error parameter identification of manipulators, and in particular observability of kinematic parameters of the 0th and nth link transformations. The relationship between error model irreducibility and commonly used observability measures is discussed in Section 4.7.6.

4.7.2 Right and Left Differential Transformations

Let $\mathbf{T}_n$ denote the homogeneous transformation matrix relating the manipulator's tool frame to the world frame,

$$\mathbf{T}_n = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2 \ldots \mathbf{A}_{n-1}\mathbf{A}_n \tag{4.356}$$

where each $\mathbf{A}_i$ is a homogeneous transformation matrix relating two coordinate frames located on two consecutive joint axes. Let q_i, $i = 1, \ldots, n$, be joint variables. This is shown in Figure 4.15 where the joint variables shown are all rotational. The -1th, 0th, and nth link coordinate frames are sometimes termed as the *world*, *base*, and *tool* frames.

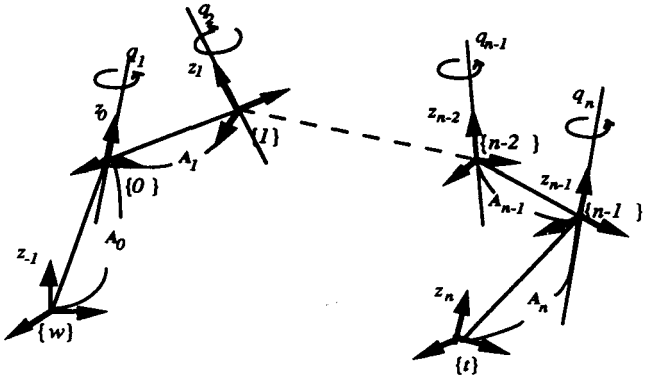


Figure 4.15. Convention for robot link coordinate frames.

Let $\mathbf{T}$ be an arbitrary homogeneous transformation matrix relating one link frame to another. We denote by $d\mathbf{T}$ the *additive differential transformation* of $\mathbf{T}$, given by

$$d\mathbf{T} = \mathbf{T} - \mathbf{T}^0 \tag{4.357}$$

where $\mathbf{T}^0$ is the transformation that corresponds to the nominal kinematic parameters and $\mathbf{T}$ corresponds to the actual kinematic parameters. The *right multiplicative differential transformation* of $\mathbf{T}$, $\Delta\mathbf{T}^u$, is defined as

$$\mathbf{T}^0\Delta\mathbf{T}^u = \mathbf{T} \tag{4.358}$$

Similarly, the *left multiplicative differential transformation* of $\mathbf{T}$, $\Delta\mathbf{T}^t$, is defined as

$$\Delta\mathbf{T}^t\mathbf{T}^0 = \mathbf{T} \tag{4.359}$$

The superscripts u and t indicate that the respective entity is associated with either the $\mathbf{U}_i$ or $\mathbf{T}_i$ matrices to be defined shortly.

The additive and multiplicative differential transformations are related through

$$\Delta\mathbf{T}^u = \mathbf{I} + (\mathbf{T}^0)^{-1}d\mathbf{T} \equiv \mathbf{I} + \delta\mathbf{T}^u \tag{4.360}$$

and

$$\Delta\mathbf{T}^t = \mathbf{I} + d\mathbf{T}(\mathbf{T}^0)^{-1} \equiv \mathbf{I} + \delta\mathbf{T}^t \tag{4.361}$$

where $\delta\mathbf{T}^u$ and $\delta\mathbf{T}^t$ have the following structure [23]:

$$\delta\mathbf{T} = \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.362}$$

where $[dx, dy, dz]^T$ are the translational errors and $[\delta x, \delta y, \delta z]^T$ are the rotational errors. This matrix structure is true for either $\delta \mathbf{T}^u$ or $\delta \mathbf{T}^t$, although values of their elements are in general different.

From Equations 4.360 and 4.361, the right and left multiplicative differential transformations are related by

$$\Delta \mathbf{T}^t = \mathbf{T}^0 \Delta \mathbf{T}^u (\mathbf{T}^0)^{-1} \tag{4.363}$$

Similarly,

$$\delta \mathbf{T}^t = \mathbf{T}^0 \delta \mathbf{T}^u (\mathbf{T}^0)^{-1} \tag{4.364}$$

To ensure that the joint rotations are about the *actual* axes after calibration, care should be exercised in selecting a differential transformation formalism. For the sake of convenience assume that all joints are revolute in the following discussion. $\mathbf{A}_i$ can be modeled in the following two alternative forms:

$$\mathbf{A}_i^0 = \text{Rot}(z, \theta_i)\mathbf{V}_i, \qquad i = 0, 1, \ldots, n \tag{4.365}$$

with $\theta_0 \equiv 0$, or

$$\mathbf{A}_i^0 = \mathbf{V}_i \,\text{Rot}(z, \theta_{i+1}), \qquad i = 0, 1, \ldots, n \tag{4.366}$$

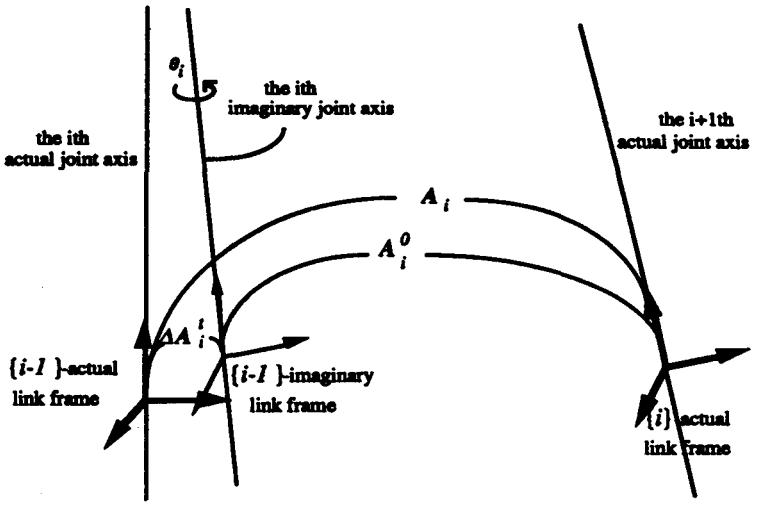
with $\theta_{n+1} \equiv 0$. In Equation 4.365 and 4.366, $\mathbf{V}_i$ is related only to the ith nominal link parameters excluding the joint variable. Assume that left multiplicative differential transformations are adopted to model kinematic errors. When $\mathbf{A}_i^0$ is modeled as in Equation 4.365,

$$\begin{aligned} \mathbf{A}_i &= \Delta \mathbf{A}_i \mathbf{A}_i^0 \\ &= \Delta \mathbf{A}_i \,\text{Rot}(z, \theta_i)\mathbf{V}_i \end{aligned} \tag{4.367}$$

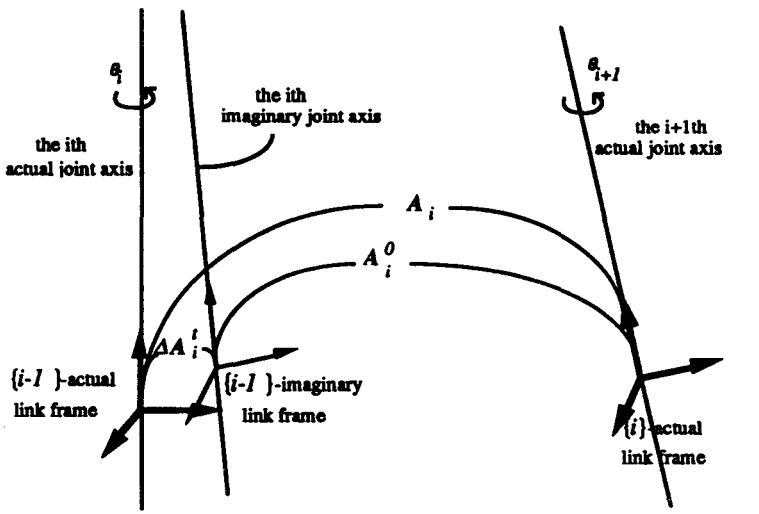
In this case, the joint variable θ_i is about the ith imaginary axis as shown in Figure 4.16a. On the other hand, when $\mathbf{A}_i^0$ is modeled as in Equation 4.366,

$$\begin{aligned} \mathbf{A}_i &= \Delta \mathbf{A}_i \mathbf{A}_i^0 \\ &= \Delta \mathbf{A}_i \mathbf{V}_i \,\text{Rot}(z, \theta_{i+1}) \end{aligned} \tag{4.368}$$

In this case, the joint variable θ_{i+1} is about the $i + 1$th actual axis as shown in Figure 4.16b. Similar analysis applies to the use of right multiplicative differential transformations. In summary, to ensure that the joint rotations are about *actual* axes after calibration, whenever $\mathbf{A}_i$ is in the form of Equation 4.366, the left multiplicative differential transformation formalism should be adopted to model kinematic errors. On the other hand, whenever $\mathbf{A}_i$ is in the form of Equation 4.365, the right multiplicative differential transformation formalism should be



(a)



(b)

Figure 4.16. Modeling kinematic errors with left multiplicative differential transformations.

chosen. The above guidelines are applicable to robots featuring prismatic joints as well.

From now on, the superscript "0" standing for "nominal" will be omitted without causing any confusion.

4.7.3 Generic Forms of Linearized Kinematic Error Models

Some preliminaries are needed for the derivations. An arbitrary 4 × 4 homogeneous transformation **T** is often written as

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{O}_{1\times 3} & 1 \end{bmatrix} \tag{4.369}$$

where **R** is a 3 × 3 orthonormal matrix and **p** is a 3 × 1 vector. A vector $\mathbf{p} = [p_x, p_y, p_z]^T$ may be represented by a skew-symmetric matrix $\mathbf{\Omega}_p$,

$$\mathbf{\Omega}_p = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \tag{4.370}$$

With this notation, Equation 4.362 can be rewritten as

$$\delta\mathbf{T} = \begin{bmatrix} \mathbf{\Omega}_\delta & \mathbf{d} \\ \mathbf{O}_{1\times 3} & 1 \end{bmatrix} \tag{4.371}$$

The following general relationship holds between a three-dimensional vector δ' and its skew-symmetric matrix representation $\mathbf{\Omega}_{\delta'}$. Let **R** be a 3 × 3 orthonormal matrix. Then

$$\mathbf{\Omega}_{\delta'} = \mathbf{R}^T\mathbf{\Omega}_\delta\mathbf{R} \tag{4.372}$$

implies that

$$\delta' = \mathbf{R}^T\delta \tag{4.373}$$

4.7.3.1 Linear Mappings Relating Cartesian Errors of an End Effector to Cartesian Errors of Individual Links

We first derive one of the mappings using right multiplicative differential transformation formalism. Let

$$\mathbf{U}_i \equiv \mathbf{A}_i\mathbf{A}_{i+1}\ldots\mathbf{A}_{n-1}\mathbf{A}_n, \qquad i = 0, 1, \ldots, n \tag{4.374}$$

Thus, $\mathbf{U}_0 = \mathbf{T}_n$ and $\mathbf{U}_{n+1} \equiv \mathbf{I}$.

A linearized kinematic error model is valid only if the kinematic model is both parametrically continuous and differentiable [33]. Parametric continuity is needed to ensure that small joint axis misalignments cause small errors in the

kinematic parameters. Differentiability of the kinematic model means that all elements of the matrix representation of the model are differentiable with respect to the link parameters.

Let $\mathbf{y}^u = [\mathbf{d}^{uT}, \boldsymbol{\delta}^{uT}]^T = [dx^u, dy^u, dz^u, \delta x^u, \delta y^u, \delta z^u]^T$ be the vector of cartesian errors of the end effector using the right multiplicative differential transformation and $\mathbf{x}^u = [(\mathbf{d}_0^u)^T, (\boldsymbol{\delta}_0^u)^T, \ldots, (\mathbf{d}_n^u)^T, (\boldsymbol{\delta}_n^u)^T]^T = [dx_0^u, dy_0^u, dz_0^u, \delta x_0^u, \delta y_0^u, \delta z_0^u, \ldots, dx_n^u, dy_n^u, dz_n^u, \delta x_n^u, \delta y_n^u, \delta z_n^u]^T$ be the vector of Cartesian errors of every link frame of the robot using the right multiplicative differential transformations.

Proposition 4.7.1: Assume that a given kinematic model $\mathbf{T}_n$ is parametrically continuous and differentiable. Then the linearized relationship between the Cartesian errors of individual links and those of the end effector is given by

$$\mathbf{y}^u = \mathbf{L}^u \mathbf{x}^u \tag{4.375}$$

where the linear mapping $\mathbf{L}^u$: $R^{6(n+1)} \to R^6$ is

$$\mathbf{L}^u = \begin{bmatrix} (\mathbf{R}_1^u)^T & (-\mathbf{R}_1^u)^T \boldsymbol{\Omega}_{p,1}^u & \cdots & (\mathbf{R}_n^u)^T & (-\mathbf{R}_n^u)^T \boldsymbol{\Omega}_{p,n}^u & \mathbf{I}_{3\times 3} & \mathbf{O}_{3\times 3} \\ \mathbf{O}_{3\times 3} & (\mathbf{R}_1^u)^T & \cdots & \mathbf{O}_{3\times 3} & (\mathbf{R}_n^u)^T & \mathbf{O}_{3\times 3} & \mathbf{I}_{3\times 3} \end{bmatrix} \tag{4.376}$$

where $(\boldsymbol{\Omega}_{p,i}^u)$, $i = 1, \ldots, n$, is a skew-symmetric matrix whose elements are $p_{i,x}^u$, $p_{i,y}^u$ and $p_{i,z}^u$, the elements of $\mathbf{p}_i^u$ associated with $\mathbf{U}_i$. The matrix $\mathbf{R}_i^u$ is the rotation matrix associated with $\mathbf{U}_i$.

Proof of Proposition 4.7.1: It is shown in [32, 33] and Equation 2.44 that

$$\delta \mathbf{T}_n^u = \sum_{i=0}^{n} \mathbf{U}_{i+1}^{-1} \delta \mathbf{A}_i^u \mathbf{U}_{i+1} \tag{4.377}$$

Then

$$\boldsymbol{\Omega}_\delta^u = \sum_{i=0}^{n} (\mathbf{R}_{i+1}^u)^T \boldsymbol{\Omega}_{\delta,i}^u \mathbf{R}_{i+1}^u \tag{4.378}$$

where $\boldsymbol{\Omega}_\delta^u$ is the upper-left 3×3 submatrix of $\delta \mathbf{T}_n^u$ and $\boldsymbol{\Omega}_{\delta,i}^u$ is the upper-left 3×3 submatrix of $\delta \mathbf{A}_i^u$. By Equations 4.372 and 4.373

$$\boldsymbol{\delta}^u = \sum_{i=0}^{n} (\mathbf{R}_{i+1}^u)^T \boldsymbol{\delta}_i^u \tag{4.379}$$

The last three rows of $\mathbf{L}^u$ in Equation 4.376 are thus obtained noting that $\mathbf{R}_{n+1}^u = \mathbf{I}_{3\times 3}$. Also from Equation 4.377,

$$\begin{aligned} \mathbf{d}^u &= \sum_{i=0}^{n} (\mathbf{R}_{i+1}^u)^T \boldsymbol{\Omega}_{\delta,i}^u \mathbf{p}_{i+1}^u + (\mathbf{R}_{i+1}^u)^T \mathbf{d}_i^u \\ &= \sum_{i=0}^{n} (-\mathbf{R}_{i+1}^u)^T \boldsymbol{\Omega}_{p,i+1}^u \boldsymbol{\delta}_i^u + (\mathbf{R}_{i+1}^u)^T \mathbf{d}_i^u \end{aligned} \tag{4.380}$$

using the fact that $\boldsymbol{\Omega}^u_{\delta,i}\mathbf{p}^u_{i+1} = \boldsymbol{\delta}^u_i \times \mathbf{p}^u_{i+1}$. The first three rows of $\mathbf{L}^u$ in Equation 4.376 are thus obtained noting that $\mathbf{R}^u_{n+1} = \mathbf{I}_{3\times 3}$ and $\boldsymbol{\Omega}^u_{p,n+1} = \mathbf{O}_{3\times 3}$. □

Derivation using left multiplicative differential transformation formalism follows a similar path. Let

$$\mathbf{T}_i = \mathbf{A}_0\mathbf{A}_1\cdots\mathbf{A}_{i-1}\mathbf{A}_i, \qquad i = 0, 1, \ldots, n \tag{4.381}$$

where $\mathbf{T}_{-1} \equiv \mathbf{I}$.

Let $\mathbf{y}^t = [\mathbf{d}^{tT}, \boldsymbol{\delta}^{tT}]^T = [dx^t, dy^t, dz^t, \delta x^t, \delta y^t, \delta z^t]^T$ be the vector of Cartesian errors of the end effector using the left multiplicative differential transformation and $\mathbf{x}^t = [(\mathbf{d}^t_0)^T, (\boldsymbol{\delta}^t_0)^T, \ldots, (\mathbf{d}^t_n)^T, (\boldsymbol{\delta}^t_n)^T]^T = [dx^t_0, dy^t_0, dz^t_0, \delta x^t_0, \delta y^t_0, \delta z^t_0, \ldots, dx^t_n, dy^t_n, dz^t_n, \delta x^t_n, \delta y^t_n, \delta z^t_n]^T$ be the vector of Cartesian errors of every link of the robot using the left multiplicative differential transformation.

Proposition 4.7.2: Assume that a given kinematic model $\mathbf{T}_n$ is parametrically continuous and differentiable. The linearized relationship between the Cartesian errors of individual links and those of the end effector is given by

$$\mathbf{y}^t = \mathbf{L}^t\mathbf{x}^t \tag{4.382}$$

where the linear mapping $\mathbf{L}^t$: $R^{6(n+1)} \to R^6$ is

$$\mathbf{L}^t = \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{O}_{3\times 3} & \mathbf{R}^t_0 & (\boldsymbol{\Omega}^t_{p,0}\mathbf{R}^t_0) & \cdots & \mathbf{R}^t_{n-1} & \boldsymbol{\Omega}^t_{p,n-1}\mathbf{R}^t_{n-1} \\ \mathbf{O}_{3\times 3} & \mathbf{I}_{3\times 3} & \mathbf{O}_{3\times 3} & \mathbf{R}^t_0 & \cdots & \mathbf{O}_{3\times 3} & \mathbf{R}^t_{n-1} \end{bmatrix} \tag{4.383}$$

Here $\boldsymbol{\Omega}^t_{p,i}$, $i = 0, \ldots, n-1$, is a skew-symmetric matrix whose elements are $p^t_{i,x}$, $p^t_{i,y}$, and $p^t_{i,z}$.

Proof: The proof is similar to that of Proposition 4.7.1. above.

The following remarks are made in cases that the matrices $\mathbf{L}^u$ and $\mathbf{L}^t$ are used as identification Jacobians. Whenever $\mathbf{A}_i$ is in the form of Equation 4.365, $\mathbf{L}^u$ should be adopted to model kinematic errors. Otherwise q_n may never appear in the identification Jacobian. Consequently, the kinematic errors associated with the nth link become unidentifiable. Similarly, whenever $\mathbf{A}_i$ is in the form of Equation 4.366, $\mathbf{L}^t$ should be chosen.

The structure of the matrices $\mathbf{L}^u$ and $\mathbf{L}^t$ is independent of the choice of a particular kinematic model. The relationship between $\mathbf{y}^u$ and $\mathbf{y}^t$ can be found using Equation 4.364. Since parallel results can be obtained using either right or left multiplicative differential transformations, the superscripts u and t are dropped in the remainder of the section whenever no confusion is caused.

4.7.3.2 Linear Mapping Relating Cartesian Errors to Link Parameter Errors

Denote by $\boldsymbol{\rho}$ the vector of kinematic parameters in a given kinematic model such that $\boldsymbol{\rho} \in \mathbf{R}^m$ and m is the number of parameters. Assuming

that the kinematic model is differentiable,

$$\mathbf{x} = \mathbf{K}d\boldsymbol{\rho} \tag{4.384}$$

where $\mathbf{K}$: $R^m \rightarrow R^{6(n+1)}$, $\mathbf{x}$ is the vector of Cartesian errors of every robot link frame, and $d\boldsymbol{\rho}$ is the parameter error vector. By Equations 4.375 and 4.382, the pose error vector $\mathbf{y}$ is related to the parameter deviations through

$$\mathbf{y} = \mathbf{L}\mathbf{K}d\boldsymbol{\rho} \tag{4.385}$$

The matrix structure of $\mathbf{K}$ depends on the particular choice of the kinematic modeling convention. For a given kinematic model, $\mathbf{K}^u$ can be derived from Equation 4.360. The matrix $\mathbf{K}^t$ can be derived from Equation 4.361. More specifically, $\mathbf{K}$ has the following form,

$$\mathbf{K} = \text{diag}(\mathbf{K}_0, \mathbf{K}_1, \ldots, \mathbf{K}_n) \tag{4.386}$$

where $\mathbf{K}_i$: $R^{m_i} \rightarrow R^{6(n+1)}$ and m_i is the number of link parameters in $\mathbf{A}_i$.

Example: A modified Denavit–Hartenberg modeling convention [13] is defined by postmultiplying $\mathbf{A}_i$ with $\text{Rot}(y, \beta_i)$, where $\mathbf{A}_i$ is as in the Denavit–Hartenberg modeling convention. In this case, the parameter vector is $[d_i, a_i, \theta_i, \alpha_i, \beta_i]^T$. The formula of $\mathbf{K}_i^u$ is as follows:

$$\mathbf{K}_i^u = \begin{bmatrix} -s\beta_i c\alpha_i & c\beta_i & a_i s\beta_i s\alpha_i & 0 & 0 \\ s\alpha_i & 0 & a_i c\alpha_i & 0 & 0 \\ c\beta_i c\alpha_i & s\beta_i & -a_i c\beta_i s\alpha_i & 0 & 0 \\ 0 & 0 & -s\beta_i c\alpha_i & c\beta_i & 0 \\ 0 & 0 & s\alpha_i & 0 & 1 \\ 0 & 0 & c\beta_i c\alpha_i & s\beta_i & 0 \end{bmatrix} \qquad i = 1, 2, \ldots, n-1 \tag{4.387}$$

Notice that the cases of $i = 0, n$ are excluded because the modified Denavit–Hartenberg convention cannot be used to model the 0th and nth link transformations [30, 33].

The matrix $\mathbf{LK}$ is an identification Jacobian of the manipulator. If a large enough number of pose measurements is taken, the identification of $d\boldsymbol{\rho}$ is possible applying least-squares methods.

4.7.4 Error Model Irreducibility

A necessary condition for the indentified Jacobian $\mathbf{LK}$ to be full rank is that the elements of $d\boldsymbol{\rho}$ are independent. If $d\boldsymbol{\rho}$ contains dependent elements, it is possible to find linear mapping relating $d\boldsymbol{\rho}$ with $\mathbf{z}$ whose elements are independent. The

problem can be stated as follows: Find a linear mapping M, where M: $R^{m'} \to R^{m}$, such that

$$d\boldsymbol{\rho} = \mathbf{M}\mathbf{z} \tag{4.388}$$

where $\mathbf{z} \in R^{m'}$; $m'(<m)$ is the number of independent parameters in the kinematic model. $\mathbf{M}$ has the following form:

$$\mathbf{M} = \text{diag}(\mathbf{M}_0, \mathbf{M}_1, \ldots, \mathbf{M}_n) \tag{4.389}$$

where $\mathbf{M}_i$: $R^{m_i'} \to R^{m_i}$; $m_{i'}$ is the number of independent parameters in $\mathbf{A}_i$. Combining Equations 4.385 and 4.386 with Equations 4.388 and 4.389 yields

$$\mathbf{KM} = \text{diag}(\mathbf{K}_0\mathbf{M}_0, \mathbf{K}_1\mathbf{M}_1, \ldots, \mathbf{K}_n\mathbf{M}_n) \tag{4.390}$$

Example: Again the modified DH model is used as an example. Whenever the $(i+1)$th joint axis is not nominally parallel to the ith axis, β_i is redundant. Thus

$$\mathbf{M}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad i = 1, 2, \ldots, n-1 \tag{4.391}$$

yielding a reduced-order parameter error vector $[dd_i, da_i, d\theta_i, d\alpha_i]^T$. Whenever the $(i+1)$th joint is parallel to the ith joint, d_i is redundant. Thus

$$\mathbf{M}_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad i = 1, 2, \ldots, n-1 \tag{4.392}$$

yielding a reduced-order parameter error vector $[da_i, d\theta_i, d\alpha_i, d\beta_i]^T$.

After $\mathbf{M}$ is found,

$$\mathbf{y} = \mathbf{LKMz} = \mathbf{Gz} \tag{4.393}$$

where $\mathbf{G} = \mathbf{LKM}$: $R^{m'} \to R^6$.

The irreducibility of a linearized error model is defined as follows:

Definition 4.7.1: Consider a parametrically continuous and differentiable kinematic model. Its linearized error model is *irreducible* if all link parameters are independent. Otherwise it is *reducible.*

The concept of irreducibility is often applied to the construction of a robust linearized error model. Model reduction is often done analytically. An alternative way of eliminating dependent parameters is through Singular Value Decomposition of the identification Jacobian.

4.7.5 Observability of Kinematic Error Parameters

Let s be the number of different measurement configurations taken to calibrate the robot. To find the inverse solution from Equation 4.393, enough measurements have to be taken. Let G_i be defined as in Equation 4.393, that is, G_i: $R^{m'} \rightarrow R^6$ for $i = 1, 2, \ldots, s$, are the identification Jacobians at each measurement configuration. Define an aggregated Jacobian matrix

$$\mathbf{G} = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_s \end{bmatrix} \tag{4.394}$$

where $\mathbf{G}$: $R^{m'} \rightarrow R^{6s}$.

It was shown [9, 10] that the number of independent parameters N in a complete model of a rigid robot must satisfy the following inequality

$$N \leq 4n - 2p + 6 \tag{4.395}$$

where p is the number of prismatic joints of the robot. A minimum number of measurement configurations can be determined accordingly. For example, 30 independent kinematic parameters are required to model a 6 degree of freedom PUMA-type robot. For a unique solution of the parameter error vector $d\boldsymbol{\rho}$, five measurement configurations must be chosen as each configuration provides six equations. If more than five measurement configurations are used, least squares methods have to be employed to solve the overdetermined system.

Definition 4.7.2: The kinematic error parameters are said to be *observable* if $G^T G$ is full rank.

The observability depends on both the kinematic modeling convention as well as the selection of measurement configurations.

Proposition 4.7.3: Assume that the right multiplicative differential transformations are used to model kinematic errors. If the orientational errors of a manipu-

lator end effector are not measured, then all the orientational parameters of $\mathbf{A}_n$ are unobservable. In this case, the number N of observable kinematic error parameters must satisfy the following inequality

$$N \leq 4n - 2p + 6 - o_n \tag{4.396}$$

where o_n is the number of independent orientational parameters in $\mathbf{A}_n$. If in addition the last joint of the manipulator is revolute and the origin of the tool frame lies on the last joint axis, then the number N of observable kinematic error parameters must satisfy the following inequality

$$N \leq 4n - 2p + 6 - o_n - ot_{n-1} \tag{4.397}$$

where $ot_{n-1} = \min\{o_{n-1}, t_{n-1}\}$; o_{n-1} and t_{n-1} are the number of independent orientational and translational parameters in $\mathbf{A}_{n-1}$, respectively.

Proof: If the orientational errors of the manipulator are not measured, then the last three rows in the linear mapping $\mathbf{L}^u$ as given in Equation 4.376 are deleted. It is clear that in this case, the columns of $\mathbf{L}^u$ corresponding to the orientational parameters of the nth link transformation are always zero. Thus the orientational parameters of $\mathbf{A}_n$ are unobservable. Combining with Equation 4.395, one has Equation 4.396.

If in addition the last joint of the manipulator is revolute and the origin of the tool frame lies on the last joint axis, $p^u_{n,x}$ and $p^u_{n,y}$ are zero and $p^u_{n,z} = p^a_{n,z}$. Thus

$$\mathbf{\Omega}^u_{p,n} = \begin{bmatrix} 0 & -p^a_{n,z} & 0 \\ p^a_{n,z} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.398}$$

$\mathbf{\Omega}^u_{p,n}$ is independent of the joint variables, therefore the columns of $(-\mathbf{R}^u_n)^T\mathbf{\Omega}^u_{p,n}$ depend linearly on the columns of $(\mathbf{R}^u_n)^T$. This implies that either the orientational parameters corresponding to the columns of $(-\mathbf{R}^u_n)^T\mathbf{\Omega}^u_{p,n}$ or the translational parameters corresponding to the columns of $(\mathbf{R}^u_n)^T$ are unobservable. Combining the result with Equation 4.396, one has Equation 4.397. □

Remark: Proposition 4.7.3 explains why only 25 parameters are independent when the PUMA arm was calibrated using only positioning errors of the end effector, as was done in references [6, 33]. Other researchers have also observed the same phenomenon [10, 31].

Proposition 4.7.4: Assume that the left multiplicative differential transformations are used to model kinematic errors. If the orientational errors of a manipulator end effector are not measured, then all orientational parameters of $\mathbf{A}_0$ are unobservable. In this case, the number N of observable kinematic error parame-

ters must satisfy the following inequality

$$N \leq 4n - 2p + 6 - o_0 \tag{4.399}$$

where o_0 is the number of independent orientational parameters in $\mathbf{A}_0$.

Proof: Similar to the proof of the first part of Proposition 4.7.3.

Proposition 4.7.5: Assume that the left multiplicative differential transformations are used to model kinematic errors. If the 0th link frame is parallel to the world frame, then the translational parameters in $\mathbf{A}_0$ are unobservable. In this case, the number N of observable kinematic error parameters must satisfy the following inequality

$$N \leq 4n - 2p + 6 - t_{01} \tag{4.400}$$

where $t_{01} = \min\{t_0, t_1\}$; t_i is the number of independent translational parameters in $\mathbf{A}_i$. If, in addition, the orientational errors of the end effector are not measured, then

$$N \leq 4n - 2p + 6 - o_0 - t_{01} \tag{4.401}$$

where o_0 is the number of independent orientational parameters in $\mathbf{A}_0$

Proof: When the base frame is parallel to the world frame, $\mathbf{R}_0^t$ is an elementary matrix. Thus $\mathbf{R}_0^t$ is independent of the joint variables (i.e., it is not a function of q_1) and the columns related to R_0^t in $\mathbf{L}^t$ depend linearly on the first three columns of $\mathbf{L}^t$. In this case, either the translational error parameters corresponding to $\mathbf{R}_0^t$ or to the first three columns of $\mathbf{L}^t$ are unobservable. Combining with Equation 4.395, one has Equation 4.400. Equation 4.401 is obtained after combining Equation 4.399 with Equation 4.400. □

Propositions 4.7.3 through 4.7.5 are true for any choice of kinematic modeling conventions. The next simple fact reveals the relationship between the observability of kinematic parameters and the irreducibility of linearized error models.

Proposition 4.7.6: Consider a parametrically continuous and differentiable kinematic model. The parameters in its linearized error model are unobservable if the linearized error model is reducible.

Proof: Denote the Jacobian matrix associated with the reducible error model as **LK** where **LK**: $R^m \to R^q$. Since the model is reducible, there exists a linear mapping **M**, where **M**: $R^{m'} \to R^m$, relating the nonindependent parameter vector to the independent parameter vector. Let $\mathbf{G} = \mathbf{LKM}$. Notice that the column rank of **LK** is the same as that of **G**. Since $m' < m$, $\text{Dim}[(\mathbf{LK})^T\mathbf{LK}] > \text{Dim}(\mathbf{G}^T\mathbf{G})$,

where Dim(·) denotes the dimension of a matrix. However Rank$[(\mathbf{LK})^T\mathbf{LK}]$ = Rank$(\mathbf{G}^T\mathbf{G})$. Thus $(\mathbf{LK})^T\mathbf{LK}$ is singular. By Definition 4.7.2 the proof is complete. □

4.7.6 Observability Measures of Reducible Error Models

Observability measures are discussed in detail in Section 4.4.

Proposition 4.7.7: Consider a parametrically continuous and differentiable kinematic model. If the linearized error model is reducible, then

$$\text{Cond}(\mathbf{G}) = \infty \tag{4.402}$$

and

$$O(\mathbf{G}) = 0 \tag{4.403}$$

Proof: If the error model is reducible, the columns of $\mathbf{G}$ are not linearly independent. In this case $\sigma_{\min} = 0$. Thus the claim holds. □

Observability measures are therefore meaningless in the case of a reducible model because in such a case, these observability measures are always zero or infinity, respectively, no matter how many measurements are taken and how well the configurations are chosen. As is well known, the matrix $\mathbf{G}^T\mathbf{G}$ can become singular at any iteration step of a numerical identification process even if the kinematic model itself has no singularities. Robust minimization techniques such as the Levenberg–Marquardt algorithm have been successfully applied to cope with the problem [3, 22]. The same techniques can provide an identification solution even if Cond$(\mathbf{G}) = \infty$ and $0(\mathbf{G}) = 0$.

4.8 CONCLUSION

In this chapter, we covered a wide range of estimation techniques that are applicable to the identification of manipulator kinematics. The purpose of the review was not only to present detailed identification algorithms, but to highlight the ideas behind and the special requirements of different methods.

Linear least-squares and nonlinear least-squares algorithms are the most straightforward and practical means for estimating the unknown kinematic parameter errors from measured data. More advanced methods such as Kalman filtering or Barker's method for robot joint axis estimation are primarily research tools intended to provide more insight into the still not fully understood relationship between the calibration error and calibration measurement accuracy.

Many interesting observability properties of robot kinematic parameter errors become evident through the study of the relationship between end effector and link Cartesian errors. It is particularly instructive to observe the role played by error modeling through either right or left differential transformations. A lot

more remains to be discovered. An important open research problem is to characterize analytically optimal sets of measurement configurations when an irreducible error model is used. This chapter described, however, several techniques that allow the designer to determine perferred robot calibration measurement configurations using intelligent off line simulation studies.

REFERENCES

[1] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[2] L. K. Barker. Vector-algebra approach to extract Denavit-Hartenberg parameters of assembled robot arms. *NASA Technology Paper 2191*, August 1983.

[3] D. J. Bennett and J. M. Hollerbach, Identifying the kinematics of robots and their tasks. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 580–586, Scottsdale, Arizona, May 1989.

[4] J. H. Borm and C. H. Menq. Experimental study of observability of parameter errors in robot calibration. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 587–592, Scottsdale, Arizona, May 1989.

[5] S. Chatterjee and B. Price. *Regression Analysis by Example.* John Wiley and Sons, 1977.

[6] J. Chen and L. M. Chao. Positioning error analysis for robot manipulators with all rotary joints. In *IEEE International Conference on Robotics and Automation*, pp. 1011–1016, April 1986.

[7] Morris R. Driels and Uday S. Pathre. Significance of observation strategy on the design of robot calibration experiments. *Journal of Robotic Systems* 7(2):197–223, June, 1990.

[8] A. Gelb (Ed.) *Applied Optimal Estimation.* MIT Press, Cambridge, MA, 1974.

[9] L. J. Everett, Morris Driels, and B. W. Mooring. Kinematic modeling for robot calibration. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, pp. 183–189, April 1987.

[10] L. J. Everett and T. W. Hsu. The theory of kinematic parameter identification for industrial robots. *ASME Journal of Dynamic Systems, Measurement, and Control*, 110(3):96–100, 1988.

[11] G. E. Forsythe and C. B. Moler. *Computer Solution of Linear Algebraic Systems.* Prentice Hall, Englewood Cliffs, NJ, 1967.

[12] A. Hald. *Statistical Theory with Engineering Applications.* John Wiley and Sons, 1952.

[13] Samad A. Hayati. Robot arm geometric link parameter estimation. In *Proceedings of the 22nd IEEE Conference on Decision and Control*, pp. 1477–1483, December 1983.

[14] IMSL. *User's Guide—FORTRAN Subroutines for Mathematical Applications, Version 1.* IMSL, Inc., 1987.

[15] V. C. Klema and A. J. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, AC-25:164–176, April 1980.

[16] K. Lau, N. Dagalakis, and D. Meyers. *Testing, International Encyclopedia of Robotics, Application, and Automation*, pp. 1753–1769. John Wiley and Sons, New York, 1988.

[17] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quarterly Applied Mathematics*, 2:164–168, 1944.

[18] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison Wesley, Reading, MA, 1973.

[19] D. W. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *Journal of the Society for Industrial Applied Mathematics*, 11(2):431–441, June 1963.

[20] J. L. Melsa and D. L. Cohn. *Decision and Estimation Theory*. McGraw-Hill, New York, 1978.

[21] C. H. Menq and J. H. Borm. Estimation and observability measure of parameter errors in a robot kinematic model. In *Proccedings of USA-Japan Symposium on Flexible Automation*, pp. 65–70, Minneapolis, Minnesota, July 1988.

[22] B. W. Mooring and S. S. Padavala. The effect of model complexity on robot accuracy. In *Proceedings of 1989 IEEE Conference on Robotics and Automation*, pp. 593–598, IEEE, May 1989.

[23] Richard P. Paul. *Robot Manipulators: Mathematics, Programming, and Control*. MIT Press, Cambridge, MA, 1981.

[24] L. E. Scales. *Introduction to Non-Linear Optimization*. Springer-Verlag, New York, 1985.

[25] M. Sklar. *Metrology and Calibration Techniques for the Performance Enhancement of Industrial Robots*. Ph.D. thesis, University of Texas at Austin, Austin, Texas, 1988.

[26] M. E. Sklar. Geometric calibration of industrial manipulators by circle point analysis. In *Proceedings of Second Conference on Recent Advances in Robotics*, pp. 178–202, Florida Atlantic University, May 1989.

[27] H. W. Stone and A. C. Sanderson. A prototype arm signature identification system. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 175–182, Raleigh, North Carolina, 1987.

[28] Henry W. Stone. *Kinematic Modeling, Identification, and Control of Robotic Manipulators*. Kluwer Academic Publishers, Boston, 1987. [Also appeared as Ph.D. dissertation, Carnegie Mellon University, 1986]

[29] Henry W. Stone, Arthur C. Sanderson, and Charles P. Neuman. Arm signature identification. In *Proceedings of 1986 IEEE International Conference on Robotics and Automation*, pp. 41–48, April 1986.

[30] W. K. Veitschegger and Chi-Haur Wu. Robot calibration and compensation. *IEEE Journal of Robotics and Automation*, 4(6):643–656, December 1988.

[31] D. E. Whitney, C. A. Lozinski, and J. M. Rourke. Industrial robot forward calibration method and results. *ASME Journal of Dynamic Systems, Measurement, and Control*, 108(1):1–8, March 1986.

[32] Chi-Haur Wu. A kinematic CAD tool for the design and control of a robot manipulator. *International Journal of Robotics Research*, 3(1):58–67, 1984.

[33] H. Zhuang. *Kinematic Modeling, Identification, and Compensation of Robot Manipulators*. Ph.D. thesis, Florida Atlantic University, December 1989.

CHAPTER 5

IMPLEMENTATION OF MANIPULATOR CALIBRATION

In the first four chapters, we concentrated on constructing an appropriate model for a given manipulator and then determining a set of coefficients that causes the model to predict the manipulator motions as accurately as possible. Once this process has been completed, the improved version of the robot model must be incorporated into the controller so that the proper relationship between workspace coordinates and joint transducer readings is achieved. This process of using the identified model to enhance the accuracy of the manipulator is referred to as the implementation step and is the focus of this chapter. It would seem that since we have developed and verified a more precise model, implementation would simply be a matter of modifying the computer algorithm in the controller. Unfortunately, this is seldom a simple process. The difficulty comes in the form of the model that has been developed.

As described in Chapters 1 and 2, most kinematic models use the joint displacements or joint transducer readings as the input and produce an estimate of the end effector pose as the output. These models are referred to as forward kinematic models. All of the model formulations described in Chapter 2 are for forward models since the joint displacements are treated as inputs. A robot controller, on the other hand, may receive a desired end effector pose as the input and must, in such a case, compute the joint displacements necessary to achieve this pose. In other words, the inverse kinematic model is required for use in the controller.

If the forward model of the manipulator is simple enough, the inverse model may be obtained by simply rearranging the equations that form the model so that the elements of the end effector pose are the inputs and the necessary joint variables are the outputs. This solution for an inverse model has been accomplished for most robots that have parallel and orthogonal axes of motion

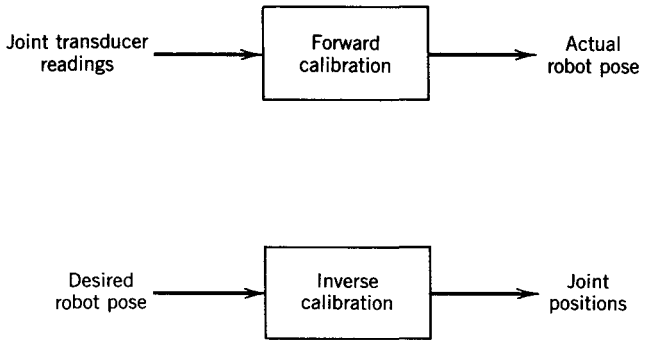


Figure 5.1. Forward and inverse calibration.

and whose wrist axes all intersect at the same point. Such robots have been termed *simple robots* by Hayati and Roston [6]. Although many robots are designed to be simple, we find that after calibration the axes are not exactly parallel or orthogonal and do not quite intersect. These variations from the design are small, but they invalidate the assumptions on which the inverse kinematic model was derived and, therefore, a new inverse kinematic model must be developed.

Following the concept of forward and inverse kinematic models, Shamma [11, 12] divided the process of robot calibration into forward and inverse calibration. As illustrated in Figure 5.1, forward calibration is defined as the process of using encoder angles or, more generally, joint transducer readings as the input to a model that will yield the actual end effector pose. Inverse calibration is defined as the reverse process. Since both the forward and inverse models are required in the controller, we have presented the calibration procedure as a set of four steps with implementation being the final part of the process. Modeling, measurement, and identification would, therefore, be forward calibration and implementation is inverse calibration.

It is important to stress that the implementation phase of robot calibration plays an important role in both off-line programmed and taught applications. In the case of off-line robot programming, which is common in advanced CAD/CAM environments, it is necessary to be able to position the end effector at the workspace coordinates of the task poses. This means that it is necessary to determine the inverse kinematics of a robot that has a nonsimple kinematic structure. Although many of today's industrial robots are nominally simple, in the sense that a simple closed-form solution exists for the nominal inverse kinematics, the conditions that create such simplicity, such as three consecutive joint axes that intersect at one point, are often lost in the actual machine. Having an identified, and presumably more exact kinematic model for the robot, therefore, cannot be considered as "end of the road." There are implementation issues that must be addressed, such as the numerical solution methods that may be used

for the inverse kinematics problem as well as the methods of modifying the robot control software.

Taught applications, in which the task points are programmed and represented in joint space, do not require the knowledge of the robot kinematic model as long as the structure of the robot does not change. For example, robot replacement on a manufacturing line without calibrating the new robot may often result in the loss of the ability to run the stored taught application. As no two robots are identical, due to unavoidable robot machining and assembly tolerances, application software normally cannot be shared between the two machines. One must either reprogram the new robot joint commands to reflect the variations in geometry or ensure that the new robot has been properly calibrated and will execute the application as precisely as the previous machine. One approach to calibration when changing robots for a taught application is to develop an algorithm to modify the joint commands at each point in the task.

Both problems, numerical inverse kinematics and joint command updating, are mathematically the same. Model-based solutions [8, 13, 14, 18–20] are discussed in detail in Sections 5.2 and 5.3. The underlying assumption behind using model-based accuracy compensation is that the prime source of accuracy errors is robot geometric errors. Accuracy compensation in the presence of significant nongeometric errors requires different methods. A "black box" approach was taken in the work by Shamma [11, 12]. The method is based on fitting of abstract interpolation functions to relate the joint transducer readings in a selected group of robot measurement configurations to measured pose errors. Such functions can then be used to compute the joint commands correction terms at the application points. Since nongeometric errors are load and configuration dependent, the requirements that such a strategy poses on the data collection phase are significantly different than the measurement requirements to accommodate model-based techniques. These issues and related archiving issues are discussed in Section 5.4.

Section 5.1 presents implementation issues on a more mathematical basis. It may be skipped, for better continuity, by readers who are interested primarily in the algorithmic side of accuracy compensation.

5.1 ACCURACY PROBLEMS IN TAUGHT AND DATA-DRIVEN APPLICATIONS

As was mentioned in the previous section it is important to distinguish between the two types of robotic tasks. Referring again to a symbolic robot kinematic model, let the model of an N-degree-of freedom robot be

$$\mathbf{x} = \mathbf{g}(\boldsymbol{\eta}, \mathbf{a}) \tag{5.1}$$

where $\mathbf{x}$ is a 6-vector of the end effector position in world coordinates (robot pose), $\boldsymbol{\eta}$ is an N-vector of joint commands (robot configuration), and $\mathbf{a}$ is a vector of all the fixed kinematic parameters.

Data-driven tasks are described in terms of a sequence (or more generally a tree) of k poses $\{\mathbf{x}(1),\ldots,\mathbf{x}(k)\}$. The number of poses, k, and the specific values $\mathbf{x}(j), j = 1,\ldots,k$, of the task points may in general depend on sensory information that is obtained on-line. For each task point $\mathbf{x}(j)$, the vector of joint commands $\boldsymbol{\eta}(j)$ must be computed through the robot inverse kinematics. The better the knowledge of the robot model, the more accurate the determination of the particular robot joint commands that will lead the robot end effector through the desired task points.

Taught tasks on the other hand are described in terms of a sequence of k configurations $\{\boldsymbol{\eta}(1),\ldots,\boldsymbol{\eta}(k)\}$ that are programmed by moving the robot manually through the task points. Ideally, taught applications do not require the knowledge of the robot kinematic model. Their success depends only on the robot's repeatability. Practically though, one needs to take into account possible changes in the robot geometry. Such effects are not necessarily slow and gradual. Permanent changes in the robot geometry may occur instantly following ordinary scheduled maintenance actions such as bearing adjustment and dismantling and replacement of the motor/bearings/encoder assembly at certain robot joints. The need to replace a robot on the manufacturing line may often result in the loss of the ability to run a stored taught application. Without calibration it is unlikely that two robots, even if they are nominally the same, will be able to share a taught application. When a robot is replaced or when permanent geometric changes occur there is a need to update the preprogrammed joint commands.

It is convenient at this point to consider two robots with similar geometries, A and B.

Let

$$\mathbf{x}_A = \mathbf{g}(\boldsymbol{\eta}_A, \mathbf{a}_A) \tag{5.2}$$

$$\mathbf{x}_B = \mathbf{g}(\boldsymbol{\eta}_B, \mathbf{a}_B) \tag{5.3}$$

be the kinematic models of robots A and B, respectively.

Robot A may represent the nominal machine. In the case of a taught application, robot A is the robot on which the task $\{\boldsymbol{\eta}_A(1),\ldots,\boldsymbol{\eta}_A(k)\}$ has been programmed. Robot B is the robot with the new geometry. Under the assumption that the kinematic identification phase has been successfully completed, the actual kinematic parameters $\mathbf{a}_B$ are assumed known. There is of course an unavoidable degree of uncertainty as no identification process is perfect. The parameter estimates $\mathbf{a}_B$ embedded within the nominal function relationship $\mathbf{g}(\cdot)$ are adopted as the new robot kinematic model parameters.

Robots A and B share the same task $\{\mathbf{x}(j), j = 1,\ldots,k\}$. Ideally it is desired that at every task point

$$\mathbf{x}_A(j) = \mathbf{x}_B(j) = \mathbf{x}(j), \qquad j = 1,\ldots,k \tag{5.4}$$

Practically, due to the imperfect nature of the identification process and the unavailable errors attributed to nongeometric parameters, some errors in accomplishing the task must be tolerated.

The first type of errors to be considered is due to the robot resolution or minimum move. The joint move resolution is often determined by one of the following factors:

1. Actuator's minimum move, as in the case of a stepping motor minimum step size.
2. Joint position sensor quantization effects, as in the case of finite word length error of an optical encoder.
3. The accuracy of the robot feedback control system at each joint, as in the case of quantization errors in a digital control system.

Mathematically such errors are described as follows: Let i denote the joint number in an N degree of freedom manipulator, $i = 1, \ldots, N$. Thus, η_i, the postion of joint i, is the ith component of the N-vector $\boldsymbol{\eta}$. Denoting the joint travel boundaries by $\eta_{i,\min}$ and $\eta_{i,\max}$, respectively, there is a finite number N_i of distinct joint commands due to the joint minimum move Δ_i.

$$N_i = \text{int}\left(\frac{\eta_{i,\max} - \eta_{i,\min}}{\Delta_i}\right) \tag{5.5}$$

Joint move resolution is rarely of any importance in taught applications. If a joint minimum move presents a serious limitation to the robot task programmer, as for instance in a component insertion application, either the position of the application fixtures need to be adjusted to fit the limitation of the given robot or the entire robot needs to be replaced by a machine that can better cope with the task requirement. In a data-driven application, on the other hand, it should be recognized that there may not exist a joint command $\boldsymbol{\eta}(j)$ that will exactly bring the robot end effector to the desired task point $\mathbf{x}(j)$. One should settle for $\boldsymbol{\eta}_A(j)$, the closest feasible neighbor of $\boldsymbol{\eta}(j)$, taking into account the given minimum joint moves. Let $\mathbf{x}_A(j)$ be the end effector position that results from implementing the command $\boldsymbol{\eta}_A(j)$. Then ε_{mmj}, the error in task step j due to the minimum-move limitation (assuming no other error sources), is

$$\varepsilon_{mmj} = \|\mathbf{x}(j) - \mathbf{x}_A(j)\|_Q \tag{5.6}$$

where $\|\cdot\|_Q$ is a convenient norm of the weighted pose error. The choice of norm as well as weighting matrix, $\mathbf{Q}$, is dictated by the nature of the task. Such norms may be selected in the following ways:

Method 1: Weighted quadratic error

$$\|\mathbf{x}(j) - \mathbf{x}_A(j)\|_{Q,2} = [\mathbf{x}(j) - \mathbf{x}_A(j)]^T \mathbf{Q} [\mathbf{x}(j) - \mathbf{x}_A(j)] \tag{5.7}$$

where $\mathbf{Q}$ is a positive semidefinite symmetric matrix. The choice of $\mathbf{Q}$ may reflect a necessary scaling action to match numerically Cartesian position errors to

certain orientation errors. **Q** may also reflect task-dependent emphasis given to particular components of the pose error vector.

Method 2: Maximum weighted absolute error

$$\|\mathbf{x}(j) - \mathbf{x}_A(j)\|_{Q,\infty} = \max\{|[\mathbf{Q}[\mathbf{x}(j) - \mathbf{x}_A(j)]]_i|, i = 1,\ldots,6\} \tag{5.8}$$

where $[\cdot]_i$ denotes the ith element of a vector. **Q** is an appropriate weighting matrix. The maximizing process runs among the absolute values of the six elements of the weighted pose error vector.

Minimum-move error is just a small factor that limits the accuracy of robots. It provides for the predictable part of the total accuracy error, unlike other error sources that are random in nature.

An important class of error sources is the one that determines robot repeatability. Repeatability has been defined as the robot's ability to return to a previously taught configuration. Imperfect repeatability is caused by

1. Clearance effects at the robot joints.
2. Disturbance and noise effect in the feedback positioning control system.
3. Gear backlash.
4. Structural deflections.

Mathematically, in a taught application, let $\boldsymbol{\eta}(j)$ denote the configuration stored in the robot controller for task step j. The actual joint positions obtained by robot A are

$$\boldsymbol{\eta}_A(j) = \boldsymbol{\eta}(j) + \boldsymbol{\varepsilon}_{\text{rep},\eta}(j) \tag{5.9}$$

where $\boldsymbol{\varepsilon}_{\text{rep},\eta}(j)$ is a random zero-mean vector. Worst case values of the error may theoretically be obtained and may be used to model the components of $\boldsymbol{\varepsilon}_{\text{rep},\eta}(j)$ as uniformly distributed random variables. These may vary from one robot pose to another due to payload and gravity loading of the robot structure. The configuration error shown in Equation 5.9 causes a robot pose error

$$\mathbf{x}(j) - \mathbf{x}_A(j) = \boldsymbol{\varepsilon}_{\text{rep},x}(j) \tag{5.10}$$

$\mathbf{x}(j)$ corresponds to the pose when teaching the values $\boldsymbol{\eta}(j)$, and $\mathbf{x}_A(j)$ is the actual pose obtained as a result of applying $\boldsymbol{\eta}_A(j)$.

Robot repeatability must also be recognized in applications programmed off-line. Taking into account repeatability effects, but assuming zero minimum-move errors and perfect knowledge of the robot kinematic model, robot total accuracy errors equal the accuracy errors due to joint repeatability. From the robot calibration point of view this represents the "ideal." Robot accuracy can never be better than robot repeatability. Minimum-move limitations and modeling errors cause the total accuracy errors $\boldsymbol{\varepsilon}_{ac}(j)$ to be larger than the "ideal"

$\varepsilon_{rep,x}(j)$. The comparison between the vectors needs to be done in a suitable vector-norm sense. Variations as large as 50:1 between $\|\varepsilon_{ac}(j)\|$ and $\|\varepsilon_{rep,x}(j)\|$ have been reported by researchers and engineers.

It may be helpful at this point to mentally isolate errors due to uncertainty in the kinematic model. Assuming perfect joint repeatability and infinitesimally small minimum move, let $\varepsilon_{mu}(j)$ denote the accuracy error in step j of the application due to model uncertainties.

$$\mathbf{x}(j) - \mathbf{x}_A(j) = \varepsilon_{mu}^{(A)}(j) \tag{5.11}$$

$$\mathbf{x}(j) - \mathbf{x}_B(j) = \varepsilon_{mu}^{(B)}(j) \tag{5.12}$$

The goal of robot calibration, in a data-driven application context, is initially to get a more accurate kinematic model of the robot. Robot B represents the robot for which a more precise model has been identified. The next objective is to improve, on the average, on the accuracy in performing every step of the task.

$$\|\varepsilon_{mu}^{(B)}(j)\| \leq \|\varepsilon_{mu}^{(A)}(j)\|, \qquad \forall j, j = 1, \ldots, k \tag{5.13}$$

The total accuracy error $\varepsilon_{ac}(j)$ is a complicated combination of the minimum-move error $\varepsilon_{mm}(j)$, the joint repeatability error $\varepsilon_{rep,x}(j)$, and the model uncertainty error $\varepsilon_{mu}(j)$. Let $M_\varepsilon(j)$ be a number that represents the maximum tolerated pose error norm in a given step j of a given robotic task. Then, robot A "needs calibration" if

$$\|\varepsilon_{ac}^{(A)}(j)\| > M_\varepsilon(j) \tag{5.14}$$

for some step $j, j = 1, \ldots, k$. The norm of $\varepsilon_{ac}^{(A)}(j)$ is taken in the same way as shown for minimum-move errors in Equation 5.7 or 5.8.

Similarly, a calibration action is declared "successful" if and only if

$$\|\varepsilon_{ac}^{(B)}\| \leq M_\varepsilon(j), \qquad \forall j, j = 1, \ldots, k \tag{5.15}$$

$\|\varepsilon_{ac}^{(B)}\|$ depends on the accuracy of identifying the robot kinematic model. This of course depends on the accuracy of the measurement equipment used in collecting the data in the calibration measurement phase.

The correction phase of robot calibration is the process of implementing the identified model of Robot B to satisfy the requirement in Equation 5.15.

In the case of a data-driven application, the correction phase is implemented (at least conceptually) by substituting the estimated kinematic parameters into the kinematic model, Equation 5.3, of robot B:

$$\boldsymbol{\mu}_B = \hat{\boldsymbol{\mu}}_B \tag{5.16}$$

$$\mathbf{a}_B = \hat{\mathbf{a}}_B \tag{5.17}$$

$$\mathbf{x}_B(j) = \mathbf{x}(j), \qquad j = 1, \ldots, k \tag{5.18}$$

where $\mathbf{x}(j)$ are the required task points. Then one determines the joint commands $\boldsymbol{\eta}_B(j), j = 1, \ldots, k$ by solving the inverse kinematics problem

$$\mathbf{x}(j) = \mathbf{g}[\boldsymbol{\eta}_B(j), \boldsymbol{\mu}_B, \mathbf{a}_B] \tag{5.19}$$

Unlike the inverse kinematics problem of robot A

$$\mathbf{x}(j) = \mathbf{g}[\boldsymbol{\eta}_A(j), \boldsymbol{\mu}_A, \mathbf{a}_A] \tag{5.20}$$

that for many commerical robots may have a simple analytic solution due to parallel or intersecting axes of motion, the inverse kinematic problem for robot B is difficult. All the simplifying assumptions that worked so well for the nominal model (robot A) are no longer valid for the actual model (robot B). The solution $\boldsymbol{\eta}_B(j)$ of Equation 5.19 is, in general, found numerically.

The correction phase for taught applications is seemingly more involved although it amounts to exactly the same difficulty as for the data-driven applications. The goal is to find a transformation method to map the task points $\boldsymbol{\eta}_A(j)$ into $\boldsymbol{\eta}_B(j)$ so that the total accuracy requirement given in Equation 5.15 is met. The software that is needed to perform such taught-task modification is part of the overall robot calibration system. The size of the correction factor that needs to be added to every joint command at every task point depends in general on the particular joint and the location within the robot work space at each given task step. It also depends on the old and new kinematic model parameters.

Updating of joint commands is sometimes done without ever using the kinematic models of robots A and B. The procedure involves limited reteaching. Robot B is moved to a few selected task points in which the joint commands $\boldsymbol{\eta}_B(j)$ (for some values of j) are recorded. Then all the rest of the task points $\boldsymbol{\eta}_B(j)$ need to be estimated using interpolation methods. The feasibility of such a technique depends on how small a portion of the total robot work space is used for the particular application. This form of joint command updating does not require any calibration measurements and any kinematic parameter identification. It is questionable whether such a method can be called "robot calibration" in the sense used in this book. The techniques shown in Section 5.4 with regard to calibration in the presence of strong nongeometric errors may be applied to implement intelligent robot reprogramming. Model-based automated robot reprogramming is also discussed in Section 5.3.

Coming back to model-based joint commands updating in taught applications, the goal is to find a formula or an algorithm for the task points transformation.

Calibration in taught applications is declared "successful" if

$$\|\mathbf{x}_B(j) - \mathbf{x}_A(j)\| \leq M_\varepsilon(j), \qquad \forall j, j = 1, \ldots, k \tag{5.21}$$

where $M_\varepsilon(j)$ is as explained in Equations 5.14 and 5.15. Rewriting Equation 5.21 in terms of the kinematic models of robots A and B

$$\|\mathbf{g}[\boldsymbol{\eta}_B(j), \mathbf{a}_B] - \mathbf{g}[\boldsymbol{\eta}_A(j), \mathbf{a}_A]\| \leq M_\varepsilon(j) \tag{5.22}$$

for every $j = 1, \ldots, k$. $\{\mathbf{a}_A\}$ are the nominal kinematic parameters, $\{\mathbf{a}_B\}$ are the identified kinematic parameters, $\boldsymbol{\eta}_A(j)$ are the taught joint commands, and $\boldsymbol{\eta}_B(j)$ are the unknown new joint commands.

The use of additive correction factors is a convenient way of updating the joint commands:

$$\boldsymbol{\eta}_B(j) = \boldsymbol{\eta}_A(j) + \Delta\boldsymbol{\eta}(j) \tag{5.23}$$

Assuming that the difference between the models of robots A and B is "small," that is

$$\|\mathbf{a}_B\| \approx \|\mathbf{a}_A\| \tag{5.24}$$

then one expects $\|\Delta\boldsymbol{\eta}(j)\|$ to be small too. One can then linearize the kinematic equations of robot B about the old set of joint commands. The following inequality is expected to hold

$$\|\mathbf{g}[\boldsymbol{\eta}_A(j), \mathbf{a}_B] + \mathbf{J}_B[\boldsymbol{\eta}_A(j)]\Delta\boldsymbol{\eta}(j) - \mathbf{g}[\boldsymbol{\eta}_A(j), \mathbf{a}_A]\| \leq M_\varepsilon(j) \tag{5.25}$$

where $\mathbf{J}_B$ is the Jacobian matrix of robot B evaluated at the task poses of robot A. Thus, at least conceptually, an approximation "formula" for the joint commands update is then given by

$$\Delta\boldsymbol{\eta}(j) = \mathbf{J}_B^{-1}[\boldsymbol{\eta}_A(j)]\{\mathbf{g}[\boldsymbol{\eta}_A(j), \mathbf{a}_B] - \mathbf{g}[\boldsymbol{\eta}_A(j), \mathbf{a}_A]\} \tag{5.26}$$

As intuitively expected, the correction factor to a first-order approximation depends on the pose error evaluated at the "old" set of configurations and on the inverse Jacobian of robot B evaluated at the task configurations of robot A.

The correction algorithm of Equation 5.26 breaks down at or near singularities of robot B. The fact that an evaluation of the inverse Jacobian of the new robot, robot B, is required makes Equation 5.26 highly impractical. This chapter explores compensation methods that use the model of robot A (rather than robot B) together with the identified kinematic parameter errors.

5.2 COMPENSATION ALGORITHMS FOR ROBOT GEOMETRIC ERRORS

5.2.1 Inverse Jacobian Based Updating of Joint Commands

Assuming that the identification phase of the calibration process has ended, the errors in the robot kinematic parameters are available. The problem of error compensation can be stated as follows:

Given:

1. The robot nominal kinematic model relating the pose homogeneous transformation matrix $\mathbf{X}$ to the vector of joint positions $\mathbf{q}$

$$\mathbf{X} = \mathbf{F}_n(\mathbf{q}) \tag{5.27}$$

2. The robot calibrated model, obtained by substituting the identified kinematic error parameters into the nominal link transformation matrices. That is, every parameter k_i is replaced by $k_i + \Delta k_i$ where Δk_i is the identified error parameter

$$\mathbf{X} = \mathbf{F}_c(\mathbf{q}) \tag{5.28}$$

3. Desired pose $\mathbf{X}_d$, and a corresponding nominal inverse kinematic solution $\mathbf{q}_n$ at this pose

$$\mathbf{q}_n = \mathbf{F}_n^{-1}(\mathbf{X}_d) \tag{5.29}$$

Find:

The necessary change $d\mathbf{q}$ of the joint positions, such that

$$\mathbf{F}_c(\mathbf{q}_n + d\mathbf{q}) = \mathbf{X}_d \tag{5.30}$$

$\mathbf{q}_n + d\mathbf{q}$ is to be used in place of $\mathbf{q}_n$ as the new set of joint commands once $d\mathbf{q}$ is determined.

A straightforward approach proposed in [13] and [8] is based on two assumptions.

Assumption 1: Positioning and orientation accuracy errors, when using the nominal kinematic model, are small.

Assumption 2: The desired pose does not fall at or near a robot singular configuration.

The algorithm works as follows. Since the accuracy error is assumed to be small, it can be regarded as a vector equal to the differential change vector of the end effector positions, when using the nominal solution $\mathbf{q}_n$ in both the nominal and the actual forward kinematics. The robot Jacobian can be used to transform this pose error vector into the corresponding joint position changes. The method must be applied iteratively until a suitable termination condition is met. An outline of the complete algorithm is given below.

Step 1: Compute an estimated robot pose $\mathbf{X}_c$ that corresponds to the available nominal inverse kinematics solution $\mathbf{q}_n$

$$\mathbf{X}_c = \mathbf{F}_c(\mathbf{q}_n) \tag{5.31}$$

Step 2: Calculate the pose error matrix between the desired pose $\mathbf{X}_d$ and the estimated actual pose $\mathbf{X}_c$.

$$d\mathbf{X} = \mathbf{X}_c - \mathbf{X}_d \tag{5.32}$$

Step 3: From $d\mathbf{X}$ form the equivalent differential error vector $d\mathbf{x}$

$$d\mathbf{x} = \begin{bmatrix} \boldsymbol{\delta} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} (\delta_x, \delta_y, \delta_z)^T \\ (d_x, d_y, d_z)^T \end{bmatrix} \tag{5.33}$$

This is done in the followng way:
One solves for the multiplicative error matrix $\boldsymbol{\Delta}$ (see Paul [10])

$$\boldsymbol{\Delta} = d\mathbf{X} \cdot \mathbf{X}_{\mathrm{d}}^{-1} \tag{5.34}$$

Ideally, the result should have the following structure

$$\boldsymbol{\Delta}_{\mathrm{id}} = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.35}$$

In most real cases, however, $d\mathbf{X}$ is not small enough. Also the actual $\boldsymbol{\Delta}$ matrix obtained from Equation 5.34 rarely has an upper 3×3 skew-symmetric block because of numerical errors. If we denote

$$\boldsymbol{\Delta} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.36}$$

then the differential error vector $d\mathbf{x}$ may be taken through averaging of off-diagonal elements of the matrix $\boldsymbol{\Delta}$.

$$\delta_x = \tfrac{1}{2}(a_{32} - a_{23}) \tag{5.37}$$

$$\delta_y = \tfrac{1}{2}(a_{13} - a_{31}) \tag{5.38}$$

$$\delta_z = \tfrac{1}{2}(a_{21} - a_{12}) \tag{5.39}$$

$$d_x = a_{14} \tag{5.40}$$

$$d_y = a_{24} \tag{5.41}$$

$$d_z = a_{34} \tag{5.42}$$

Step 4: Compute changes in joint positions using

$$d\mathbf{q} = \mathbf{J}^{-1} d\mathbf{x} \tag{5.43}$$

where $\mathbf{J}$ is the robot Jacobian formulated using the *nominal* model.

Step 5: Update joint commands by setting:

$$\mathbf{q} = \mathbf{q}_n + d\mathbf{q} \tag{5.44}$$

Step 6: Check an appropriate termination condition. For instance, if at least one element of $d\mathbf{q}$ becomes smaller than the resolution of the corresponding robot joint position encoder, then terminate the algorithm.

Another suitable termination condition may be based on comparing $\|d\mathbf{q}^{(k)} - d\mathbf{q}^{(k-1)}\|$ to a desired lower threshold. The integer k denotes the current iteration. If the termination condition is not met, the six-step procedure is repeated.

The computational efficiency of this algorithm depends critically on Equation 5.29 and Step 4. In fact, for robots with general geometries, finding the compensated joint commands with the foregoing algorithm may not be any more effective than directly solving the inverse kinematics of the calibrated robot using some numerical method. For robots with relatively simple geometries, such as is the case with most industrial robots, much simplification on computation can usually be obtained. In Equation 5.29, for example, closed-form solutions have been determined for robots containing a wrist center, which is almost invariably the case with nominal industrial robots. Moreover, as has been demonstrated in [15], with proper formulation, inversion of the Jacobian for similar geometrically simple types of robots can be obtained analytically in closed form. Note that in Step 4 of this approach the Jacobian is formulated based on the nominal model. This allows for the algorithm to exploit the full advantage of closed-form solutions that are typically available for most industrial robots. The result is, of course, a significant reduction of the computations involved.

A general and systematic method of developing the Jacobian inverse analytically has been proposed by Waldron et al. [15]. The main idea of their method is to formulate the Jacobian by selecting an appropriate reference frame in which the Jacobian is expressed and the point of reference to which the end effector velocity is referred. It suggests that if the robot geometry contains concurrent joint axes, using a reference frame with the origin at the point of concurrency will greatly simplify the Jacobian. If the geometry includes parallel joint axes, the Jacobian can likewise be simplified by transforming to a reference frame parallel to those axes.

For example, given robots with axes 4, 5, and 6 intersecting, such as a PUMA 560, an effective choice of the reference frame is on link 3 with its origin at the point of concurrency (the wrist center). This places the reference point of end effector velocity at the point that is instantaneously coincident with the origin of the fixed reference frame. The Jacobian that results will, in general, be of the following form:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{0} \end{bmatrix} \tag{5.45}$$

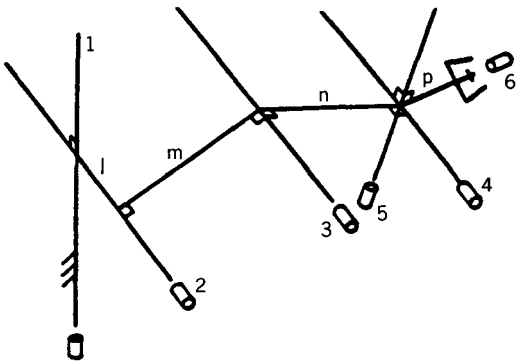


Figure 5.2. Geometry of manipulator chain of example 1. Adapted from reference [15].

Note that all the elements in the lower right-hand quadrant are zero. This is a direct consequence of the axes 4, 5, 6 of the robot being concurrent. It can be seen that analytic inversion of the Jacobian becomes feasible, since, in terms of the application here, the lower three components of $d\mathbf{q}$ are now functions of the lower three translational error components of $d\mathbf{x}$, which are directly solvable. The remaining rotational error (upper three) components of $d\mathbf{q}$ can be obtained subsequently by treating the already found components as knowns. Of course, the differential error vector, $d\mathbf{x}$, must be transformed into the same reference frame, frame 3 in this example, in which the Jacobian is expressed.

Example 1: Simulation results of an elbow manipulator.

Referring to Figure 5.2, which shows the nominal geometry of the robot, Table 5.1 summarizes the Denavit–Hartenberg (DH) parameters for the nominal model.

The length parameter values were adopted from the PUMA 560 nominal model as given in [5], although the two nominal models are not quite the same.

In the work by Waldron et al. [15], explicit formulas for the manipulator Jacobian that relates the vector of joint rates to a particular choice of a world-coordinates velocity vector $\boldsymbol{\mu}$ are derived.

TABLE 5.1. DH Parameters of the Simulated Elbow Manipulator

Link Number	a_i(mm)	d_i(mm)	α_i(degrees)	θ_i
1	0	0	90	Variable
2	$m = 431.8$	$l = 149.09$	0	Variable
3	$n = 433.07$	0	0	Variable
4	0	0	90	Variable
5	0	0	90	Variable
6	0	$p = 56.25$	0	Variable

The resulting inverse Jacobian formulas led to the following joint command uppdates:

$$d\theta_1 = \frac{-\mu_z}{mc\theta_2 + nc(\theta_2 + \theta_3)} \tag{5.46}$$

$$d\theta_2 = \frac{1}{ms\theta_3}[\mu_x - (d\theta_1)lc(\theta_2 + \theta_3)] \tag{5.47}$$

$$d\theta_3 = \frac{1}{n}[(d\theta_1)ls(\theta_2 + \theta_3) - (d\theta_2)(n + mc\theta_3) + \mu_y] \tag{5.48}$$

$$d\theta_5 = \delta_x s\theta_4 - \delta_y c\theta_4 + (d\theta_1)c(\theta_2 + \theta_3 + \theta_4) \tag{5.49}$$

$$d\theta_6 = \frac{1}{s\theta_5}[\delta_x c\theta_4 + \delta_y s\theta_4 - (d\theta_1)s(\theta_2 + \theta_3 + \theta_4)] \tag{5.50}$$

$$d\theta_4 = \delta_z - (d\theta_2) - (d\theta_3) + (d\theta_6)c\theta_5 \tag{5.51}$$

where

$$\boldsymbol{\mu} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} - \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} \times \begin{bmatrix} pc\theta_4 s\theta_5 \\ ps\theta_4 s\theta_5 \\ -pc\theta_5 \end{bmatrix} \tag{5.52}$$

Equations 5.46 through 5.52 were used in simulating the compensation algorithm for various "calibrated" models of the elbow manipulator as described by Huang and Gautam [8], in certain designated desired robot configurations. Nominal inverse kinematic solutions for the PUMA 560 robot are available in the work by Fu et al. [5], and versions of these were adopted here.

The rate of convergence for finding the solutions, using the joint resolution criterion, was found to vary from four to six iterations in all cases tested. Some typical simulation results for the Elbow robot are presented in Table 5.2. It can be seen that the accuracy is improved by approximately a factor of 40 to 1 after compensation. Note that in the simulation the errors introduced to the robot model are much worse than can be seen in real robots—the error for a PUMA

TABLE 5.2. Typical Simulation Results for Elbow Robot

	X Axis Error		*Y* Axis Error		*Z* Axis Error		Total RMS Error	
No.	Initial	Comp.	Initial	Comp.	Initial	Comp.	Initial	Comp.
1	5.59	0.04	2.97	0.17	2.55	0.07	6.82	0.19
2	2.64	0.00	3.08	0.13	4.25	0.10	5.88	0.16
3	2.67	0.08	4.31	0.09	2.18	0.04	5.52	0.13

is typically less than 5 mm in practice, thus it is reasonable to expect that an even better rate of convergence can be obtained.

The number of operations required to calculate the $d\mathbf{x}$ vector are three multiplications, nine additions, and no transcendental function calls ($3M + 9A + 0F$). The transformation of $d\mathbf{x}$ into frame 3 requires 24 multiplications, 27 additions, and no function calls ($24M + 27A + 0F$). The calculation of $d\mathbf{q}$ using the analytic solution of Jacobian inverse requires 24 multiplications, 16 additions, and 8 function calls ($24M + 16A + 8F$). The total number of operations required for each iteration of the compensation scheme are 51 multiplications, 52 additions, and 8 function calls ($51M + 52A + 8F$). Although the number of iterations is more than that claimed in [13], it should be noted that the computational effort required in each iteration is much smaller as the result of the use of analytic closed-form kinematic solutions.

Similar strategy was proposed in [13]. All inverse-Jacobian-based algorithms may not converge if the kinematic parameter errors are large. At singular configurations the algorithm breaks down since the inverse Jacobian does not exist. It is a common practice not to teach task points that are at or near the robot singular configurations. To ensure that no entry to the singular zones is encountered during the iterations of the compensation algorithm sufficient safety margins from the singular zones need to be maintained while the robot is programmed.

5.2.2 Redefinition of Task Points Displacement Matrices

The key idea is to redefine the desired positions and orientations in amounts that will create an equivalent effect to that of updating the robot joint commands. This was suggested originally by Veitchegger and Wu [13] and brought into a more complete form by Vuskovic [14].

Let

$$\mathbf{X}_d = \begin{bmatrix} \mathbf{R}_d & \mathbf{p}_d \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.53}$$

where $\mathbf{p}_d$ and $\mathbf{R}_d$ are the desired end effector position vector and orientation matrix, respectively. Let $\Delta\mathbf{p}_d$ and Λ_d be the position and orientation modifications. Thus, the modified desired pose $\mathbf{X}_m$ is

$$\mathbf{X}_m = \begin{bmatrix} (\mathbf{I} + \Lambda_d)\mathbf{R}_d & \mathbf{p}_d + \Delta\mathbf{p}_d \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.54}$$

The modified desired pose to the robot nominal inverse kinematics is used to produce joint commands that will move the actual manipulator to the originally desired pose, as follows (see Eq. 5.27–5.28):

$$\mathbf{F}_c[\mathbf{F}_n^{-1}(\mathbf{X}_m)] = \mathbf{X}_d \tag{5.55}$$

The problem is to solve for the appropriate $\Delta\mathbf{p}_d$ and Λ_d that will satisfy Equation 5.55. $\Lambda_d = \Lambda_d(\Delta\mathbf{k}_d)$ is a skew-symmetric matrix associated with the vector $\Delta\mathbf{k}_d$ of orthogonal rotations of the last link about the basis vectors of the manipulator base.

The term $\Delta\mathbf{k}_d$ is found similarly to $\boldsymbol{\delta}$ of Equation 5.33. Denoting the manipulator's end effector position vector $\mathbf{x}$ in terms of $\mathbf{p}$ and a vector $\mathbf{k}$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{k} \\ \mathbf{p} \end{bmatrix} = \mathbf{g}(\mathbf{q}, \mathbf{a}) \tag{5.56}$$

where $\mathbf{q}$ is the vector of joint positions and $\mathbf{a}$ is the vector of fixed kinematic parameters. It is shown by Vuskovic [14] via Taylor series expansion of Equation 5.55 that a linearized solution for $\Delta\mathbf{p}_d$ and $\Delta\mathbf{k}_d$ is

$$\Delta\mathbf{p}_d = -\frac{\partial\mathbf{p}}{\partial\mathbf{a}}\Delta\mathbf{a} \tag{5.57}$$

$$\Delta\mathbf{k}_d = -\frac{\partial\mathbf{k}}{\partial\mathbf{a}}\Delta\mathbf{a} \tag{5.58}$$

where $\Delta\mathbf{a}$ is the identified vector of kinematic parameter errors. The terms $\partial\mathbf{p}/\partial\mathbf{a}$ and $\partial\mathbf{k}/\partial\mathbf{a}$ are "kinematic sensitivities" that constitute the identification Jacobian derived in Chapter 2.

The kinematic sensitivity matrices can be computed explicitly as shown in [14] using geometric interpretation of Jacobian matrices as originally suggested by Whitney [16] as follows:

$$\frac{\partial\mathbf{p}}{\partial a_i} = \mathbf{x}_i \tag{5.59}$$

$$\frac{\partial\mathbf{p}}{\partial\alpha_i} = \mathbf{x}_i \times \mathbf{r}_i \tag{5.60}$$

$$\frac{\partial\mathbf{p}}{\partial d_i} = \mathbf{z}_i \tag{5.61}$$

$$\frac{\partial\mathbf{p}}{\partial\theta_i} = \mathbf{z}_i \times \mathbf{r}_i \tag{5.62}$$

$$\frac{\partial\mathbf{p}}{\partial\beta_i} = \mathbf{y}_i \times (\mathbf{r}_i - a_i\mathbf{x}_i) \tag{5.63}$$

where a_i, α_i, d_i, θ_i are the DH parameters and β_i is the additional kinematic parameter that may be added to handle nominally parallel joint axes (see Chapter 2). $\mathbf{x}_i$, $\mathbf{y}_i$, $\mathbf{z}_i$ are the basis vectors of the ith link where the link coordinate assignments are done according to Craig's modification of the DH convention

[3] such that the axis $\mathbf{z}_i$ is colinear with the ith joint axis. $\mathbf{r}_i$ is the vector connecting the last link frame origin with the ith link frame origin.

Similarly:

$$\frac{\partial \mathbf{k}}{\partial a_i} = \mathbf{0} \tag{5.64}$$

$$\frac{\partial \mathbf{k}}{\partial \alpha_i} = \mathbf{x}_i \tag{5.65}$$

$$\frac{\partial \mathbf{k}}{\partial d_i} = \mathbf{0} \tag{5.66}$$

$$\frac{\partial \mathbf{k}}{\partial \theta_i} = \mathbf{z}_i \tag{5.67}$$

$$\frac{\partial \mathbf{k}}{\partial \beta_i} = \mathbf{y}_i \tag{5.68}$$

The vectors $\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i, \mathbf{r}_i$ can be computed recursively for simple nominal manipulator models, as follows:

If $\alpha_{i-1} = 0$, then

$$\mathbf{x}_i = (s\theta_i)\mathbf{y}_{i-1} + (c\theta_i)\mathbf{x}_{i-1} \tag{5.69}$$

$$\mathbf{y}_i = (c\theta_i)\mathbf{y}_{i-1} + (s\theta_i)\mathbf{x}_{i-1} \tag{5.70}$$

$$\mathbf{z}_i = \mathbf{z}_{i-1} \tag{5.71}$$

If $\alpha_{i-1} = \pm 90°$, then

$$\mathbf{x}_i = \pm(s\theta_i)\mathbf{z}_{i-1} + (c\theta_i)\mathbf{x}_{i-1} \tag{5.72}$$

$$\mathbf{y}_i = \pm(c\theta_i)\mathbf{z}_{i-1} + (s\theta_i)\mathbf{x}_{i-1} \tag{5.73}$$

$$\mathbf{z}_i = \pm\mathbf{y}_{i-1} \tag{5.74}$$

where

$$\mathbf{x}_0 = (1, 0, 0)^T \tag{5.75}$$

$$\mathbf{y}_0 = (0, 1, 0)^T \tag{5.76}$$

$$\mathbf{z}_0 = (0, 0, 1)^T \tag{5.77}$$

Finally:

$$\mathbf{r}_i = \mathbf{r}_{i+1} + a_i\mathbf{x}_i + d_{i+1}\mathbf{z}_{i+1} \tag{5.78}$$

$$\mathbf{r}_N = 0 \tag{5.79}$$

Substitution of Equations 5.59 through 5.79 into Equations 5.57 through 5.58 yields the general formulas for desired pose compensation that hold for any N-degrees-of-freedom open-chain manipulator:

$$\Delta \mathbf{p}_d = -\sum_{i=1}^{N} (\mathbf{x}_{i-1}\Delta a_{i-1} + \mathbf{z}_i \Delta d_i) - \sum_{i=1}^{N} (\mathbf{x}_{i-1}\Delta\alpha_{i-1}) \times \mathbf{r}_{i-1}$$
$$-\sum_{i=1}^{N} (\mathbf{y}_{i-1}\Delta\beta_{i-1}) \times (\mathbf{r}_{i-1} - a_{i-1}\mathbf{x}_{i-1}) \tag{5.80}$$

$$\Delta \mathbf{k}_d = -\sum_{i=1}^{N} (\mathbf{x}_i \Delta\alpha_i + \mathbf{y}_i \Delta\beta_i) \tag{5.81}$$

Example 2: Vuskovic's pose compensation formulas [14] for a PUMA manipulator.

The above formulas are specialized to the particular set of nominal link parameter values of the PUMA manipulator. These are

$$\alpha_0 = \alpha_2 = 0$$
$$\alpha_1 = \alpha_3 = \alpha_5 = -90^\circ$$
$$\alpha_4 = 90^\circ$$
$$a_0 = a_1 = a_4 = a_5 = 0$$
$$d_1 = d_2 = d_5 = d_6 = 0$$
$$r_0 = r_1 = r_2 = p$$
$$r_4 = r_5 = r_6 = 0$$

Upon obtaining the calibrated kinematic parameter errors, the following entities may be computed:

$$\zeta_1 = (c\theta_2)\Delta\alpha_2 - (s\theta_2)\Delta\beta_2 \qquad \eta_1 = (s\theta_2)\Delta\alpha_2 - (c\theta_2)\Delta\beta_2$$
$$\zeta_2 = \zeta_1 + c(\theta_2 + \theta_3)\Delta\alpha_3 + \Delta\alpha_1 \qquad \eta_2 = -s(\theta_2 + \theta_3)\Delta\alpha_3 - \eta_1$$
$$\zeta_3 = \Delta a_4 + (c\theta_5)\Delta a_5 \qquad \eta_3 = d_3\Delta a_3 + d_4 + (s\theta_5)\Delta a_5$$
$$\zeta_4 = (c\theta_4)\Delta d_5 - (s\theta_4)\zeta_3 \qquad \eta_4 = (s\theta_4)\Delta d_5 - (c\theta_4)\zeta_3$$
$$\zeta_5 = \zeta_4 + (s\theta_3)a_2\Delta\alpha_3 + \Delta d_2 + \Delta d_3 + a_2\Delta\beta_2 \qquad \eta_5 = \eta_4 + (s\theta_1)\Delta a_3$$
$$\zeta_6 = c(\theta_2 + \theta_3)\eta_5 - s(\theta_2 + \theta_3)\eta_3 \qquad \eta_6 = s(\theta_2 + \theta_3)\eta_5 + c(\theta_2 + \theta_3)\eta_3$$
$$\zeta_7 = \zeta_6 + (c\theta_2)\Delta a_2 + \Delta a_1 \qquad \eta_7 = -(s\theta_2)\Delta a_2 + \Delta d_1$$
$$\zeta_8 = (c\theta_1)\Delta\zeta_7 - (s\theta_1)\zeta_5 \qquad \eta_8 = (s\theta_1)\Delta\zeta_7 + (c\theta_1)\zeta_5$$
$$\zeta_9 = \alpha_4 + (c\theta_2)\Delta\alpha_5 \qquad \eta_9 = (s\theta_5)\Delta\alpha_5$$
$$\zeta_{10} = (c\theta_4)\zeta_9 \qquad \eta_{10} = (s\theta_4)\zeta_9$$
$$\zeta_{11} = c(\theta_2 + \theta_3)\zeta_{10} - s(\theta_2 + \theta_3)\eta_9 \qquad \eta_{11} = s(\theta_2 + \theta_3)\zeta_{10} + c(\theta_2 + \theta_3)\eta_9$$
$$\zeta_{12} = (c\theta_1)\eta_{10} - (s\theta_1)\zeta_{11} \qquad \eta_{12} = (s\theta_1)\eta_{10} + (c\theta_1)\zeta_{11}$$

Then

$$\Delta \mathbf{k}_d = \begin{bmatrix} -(c\theta_1)\zeta_2 - \Delta\alpha_0 + \eta_{12} \\ -(s\theta_1)\zeta_2 - \zeta_{12} \\ -\eta_2 - \eta_{11} \end{bmatrix}$$

and

$$\Delta \mathbf{p}_d = \begin{bmatrix} -(c\theta_1)\zeta_2 - \Delta\alpha_0 \\ -(s\theta_1)\zeta_2 \\ -\eta_2 \end{bmatrix} \times \mathbf{p}_d - \begin{bmatrix} \zeta_8 + \Delta a_0 \\ \eta_8 \\ \eta_7 \end{bmatrix} - \mathbf{a}_d \Delta d_6$$

where $\mathbf{a}_d$ is the desired approach vector of the wrist (i.e., the third column of the desired orientation matrix $\mathbf{R}_d$).

As claimed in [14], simulation studies show that the method appears to be working well even at singular configurations.

5.3 OPTIMAL DESIGN OF ROBOT ACCURACY COMPENSATORS

The algorithms for robot accuracy compensation discussed in the previous section were derived using task-point attainment equality conditions, that is, assuming that the desired robot pose after compensation can be attained without error. At this point it is worthwhile to realize that several sources still contribute to the robot pose error even in the presence of accuracy compensation. Among these are

1. Imperfect identification of the robot kinematic parameter errors. Thus, the actual kinematic model contains uncertainties.
2. All the compensation algorithms that have been discussed thus far are based on linearized error models.
3. Existence of unmodeled nongeometric errors.

In this section, the problem of joint commands updating using the identified kinematic error parameters is cast as an optimization problem as demonstrated in the work by Zhuang et al. [17–20].

Several design tradeoffs are highlighted while optimizing the solution. The first tradeoff is between the attained compensated pose error and the size of the correction vector to the robot joint commands. Obtaining sufficiently small accuracy errors may sometimes require exceedingly large joint command changes. If this may result in hitting a joint travel limit, or in entering a robot singular zone, then, of course, the compensation action becomes impaired.

The second tradeoff is between end effector position and orientation errors. Different applications may emphasize different aspects of robot accuracy. There are also scaling problems that may impair the quality of the compensation. Positioning errors, that are absolute errors, sometimes have orders of magnitude difference with respect to orientation errors. Thus, the weighting matrix that is used as part of the optimization performance measure may be adjusted to avoid numerically difficult scaling differences.

Like the methods discussed previously, the compensation method discussed here is also based on a linearization of the kinematic model and as such is valid only if the identified errors are sufficiently small. There is no need to solve explicitly the inverse kinematics problem of the actual robot. The correction algorithm is linear in terms of the identified kinematic parameter errors. The coefficients of the kinematic errors that affect the compensation algorithm depend on the nominal kinematic model and the nominal task points only. Because of the particular choice of performance index, the solution exists at all robot poses. For example, near singularities of the nominal model, the optimal strategy may be to introduce no corrections. The size of corrections can be made to depend on the distance of the task point from the robot workspace boundaries or special singularity zones.

5.3.1 Mathematical Background—Linear Quadratic Regulators for First-Order Discrete Time Systems

Consider the following particular case of a first-order linear discrete-time system:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{b}_k u_k, \qquad k = 1, 2, \ldots \tag{5.82}$$

where $\mathbf{x}_k$, the state vector, and $\mathbf{b}_k$ the vector coefficient of the scalar control signal u_k, are both n-vectors. The system is time varying as $\mathbf{b}_k$ does not necessarily retain the same values from one time instant k to another. It is assumed that the vector $\mathbf{b}_k$ is known at every $k = 1, 2, \ldots$ and so is the initial state vector $\mathbf{x}_1$.

The optimal control problem is to find a set of N numbers u_k, $k = 1, \ldots, N$ that are the control values such that the following performance measure $J(u_1, \ldots, u_N)$ is minimized:

$$J(u_1, \ldots, u_N) = \mathbf{x}_{N+1}^T \mathbf{Q} \mathbf{x}_{N+1} + \sum_{k=1}^{N} \gamma_k u_k^2 \tag{5.83}$$

where $\mathbf{Q}$ is a given symmetric nonnegative definite matrix. Also given are the numbers γ_k, $k = 1, \ldots, N$ assumed to be strictly positive.

In simple words, the objective of the control problem is to drive the state vector $\mathbf{x}_1$ in N steps as close as possible to zero without using excessive control values.

The main result is the following recursive solution algorithm:

Theorem: The optimal control values u_k^*, $k = 1, \ldots, N$ that minimize $J(u_1, \ldots, u_N)$ of Equation 5.83 subject to Equation 5.82 are expressed in the following control law:

$$u_k^* = \mathbf{k}_k^T \mathbf{x}_k, \qquad k = 1, \ldots, N \tag{5.84}$$

where the row vector of gain coefficients $\mathbf{k}_k^T$ is obtained recursively as follows:

$$\mathbf{k}_{N-k}^T = -\frac{\mathbf{b}_{N-k}^T \mathbf{P}_k}{\mathbf{b}_{N-k}^T \mathbf{P}_k \mathbf{b}_{N-k} + \gamma_{N-k}}, \qquad k = 0, 1, \ldots, N-1 \tag{5.85}$$

$$\mathbf{P}_{k+1} = \mathbf{P}_k + \mathbf{P}_k \mathbf{b}_{N-k} \mathbf{k}_{N-k}^T \tag{5.86}$$

$$\mathbf{P}_0 = \mathbf{Q} \tag{5.87}$$

The resulting minimum cost J^* is

$$J^* = J(u_1^* \ldots, u_N^*) = \mathbf{x}_1^T \mathbf{P}_N \mathbf{x}_1 \tag{5.88}$$

Proof: A standard dynamic programming argument is to assume that optimal controls $[u_1^*, \ldots, u_{N-1}^*]$ have already been selected and the problem is to find u_N^* at the final step. Let $J_{N,N+1}$ be the cost function for optimizing u_N, where

$$J_{N,N+1} = \mathbf{x}_{N+1}^T Q \mathbf{x}_{N+1} + \gamma_N u_N^2 \tag{5.89}$$

Denote $\mathbf{P}_0 = \mathbf{Q}$. By substituting from Equation 5.82 for $\mathbf{x}_{N+1}$ one gets

$$J_{N,N+1} = (\mathbf{x}_N + \mathbf{b}_N u_N)^T \mathbf{P}_0 (\mathbf{x}_N + \mathbf{b}_N u_N) + \gamma_N u_N^2 \tag{5.90}$$

By differentiating $J_{N,N+1}$ with respect to u_N

$$\frac{\partial J_{N,N+1}}{\partial u_N} = 2\gamma_N u_N + 2\mathbf{b}_N^T \mathbf{P}_0 (\mathbf{x}_N + \mathbf{b}_N u_N) \tag{5.91}$$

and equating to zero u_N^* is found:

$$u_N^* = -\frac{\mathbf{b}_N^T \mathbf{P}_0}{\mathbf{b}_N^T \mathbf{P}_0 \mathbf{b}_N + \gamma_N} \mathbf{x}_N \tag{5.92}$$

Since the second derivative of $J_{N,N+1}$ with respect to u_N is positive, u_N^* is the global minimum. Denote

$$\mathbf{k}_N^T = \frac{-\mathbf{b}_N^T \mathbf{P}_0}{\mathbf{b}_N^T \mathbf{P}_0 \mathbf{b}_N + \gamma_N} \tag{5.93}$$

then

$$u_N^* = \mathbf{k}_N^T \mathbf{x}_N \tag{5.94}$$

The optimal cost at the final step is then

$$J_{N,N+1}^* = \mathbf{x}_N^T \{\mathbf{I} + \mathbf{b}_N \mathbf{k}_N^T)^T \mathbf{P}_0 (\mathbf{I} + \mathbf{b}_N \mathbf{k}_N^T) + \mathbf{k}_N \gamma_N \mathbf{k}_N^T\} \mathbf{x}_N \tag{5.95}$$

$$= \mathbf{x}_N^T \left\{ \mathbf{P}_0 - \frac{\mathbf{P}_0 \mathbf{b}_N \mathbf{b}_N^T \mathbf{P}_0}{\mathbf{b}_N^T \mathbf{P}_0 \mathbf{b}_N + \gamma_N} \right\} \mathbf{x}_N \tag{5.96}$$

$$= \mathbf{x}_N^T \{\mathbf{P}_0 + \mathbf{P}_0 \mathbf{b}_N \mathbf{k}_N^T\} \mathbf{x}_N \tag{5.97}$$

$$= \mathbf{x}_N^T \mathbf{P}_1 \mathbf{x}_N \tag{5.98}$$

where

$$\mathbf{P}_1 = \mathbf{P}_0 + \mathbf{P}_0 \mathbf{b}_N \mathbf{k}_N^T \tag{5.99}$$

The same strategy is now repeated to find u_{N-1}^*,

$$u_{N-1}^* = \mathbf{k}_{N-1}^T \mathbf{x}_{N-1} \tag{5.100}$$

where

$$\mathbf{k}_{N-1}^T = -\frac{\mathbf{b}_{N-1}^T \mathbf{P}_1}{\mathbf{b}_{N-1}^T \mathbf{P}_1 \mathbf{b}_{N-1} + \gamma_{N-1}} \tag{5.101}$$

Let $J_{N-1,N+1} = J_{N-1,N} + J_{N,N+1}$ be the cumulative cost for the last two steps, then

$$J_{N-1,N+1}^* = \mathbf{x}_{N-1}^T \mathbf{P}_2 \mathbf{x}_{N-1} \tag{5.102}$$

where

$$\mathbf{P}_2 = \mathbf{P}_1 + \mathbf{P}_1 \mathbf{b}_{N-1} \mathbf{k}_{N-1}^T \tag{5.103}$$

By mathematical induction, for the kth stage

$$u_{N-k}^* = \mathbf{k}_{N-k}^T \mathbf{x}_{N-k} \tag{5.104}$$

where

$$\mathbf{k}_{N-k}^T = -\frac{\mathbf{b}_{N-k}^T \mathbf{P}_k}{\mathbf{b}_{N-k}^T \mathbf{P}_k \mathbf{b}_{N-k} + \gamma_{N-k}} \tag{5.105}$$

and

$$J_{N-k,N+1}^* = \mathbf{x}_{N-k}^T \mathbf{P}_k \mathbf{x}_{N-k} \tag{5.106}$$

where

$$\mathbf{P}_{k+1} = \mathbf{P}_k + \mathbf{P}_k \mathbf{b}_{N-k} \mathbf{k}_{N-k}^T \tag{5.107}$$

Finally, the minimum cost is

$$J_{1,N+1}^* = x_1^T \mathbf{P}_N \mathbf{x}_1 \tag{5.108}$$

That completes the proof.

Studying the solution, Equation 5.85 reveals that the requirement that all coefficients γ_k be strictly positive is intended to ensure that u_k^* never attains infinite values as might happen for particular values of $\mathbf{b}_k$ should the performance criterion in Equation 5.83 be based on the terminal error only.

5.3.2 Joint Command Updating Problem Formulation as an Optimal Control Problem

The position and orientation of the end effector in world coordinates of an N degree-of-freedom manipulator can be represented by the following 4×4 homogeneous transformation matrix $\mathbf{T}_N$

$$\mathbf{T}_N = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.109}$$

We define a vector $\mathbf{f} = [f_1, f_2, \ldots, f_{12}]^T$, consisting of the nontrivial elements of $\mathbf{T}_N$, that is, $[f_1, f_2, f_3]^T = \mathbf{n} = [n_x, n_y, n_z]^T$, $[f_4, f_5, f_6] = \mathbf{o} = [o_x, o_y, o_z]^T$, $[f_7, f_8, f_9] = \mathbf{a} = [a_x, a_y, a_z]^T$ and $[f_{10}, f_{11}, f_{12}]^T = \mathbf{p} = [p_x, p_y, p_z]^T$.

Accuracy errors may be modeled in terms of an error matrix $\Delta\mathbf{T}_N$, or equivalently an error vector $\Delta\mathbf{f}$. The problem is to find adjustments of the joint variables such that $\Delta\mathbf{f}$ is minimized (of course in a certain norm sense, $\|\Delta\mathbf{f}\|$).

The 12-vector $\mathbf{f}$ is the function of the vector of link parameters denoted here by $\boldsymbol{\rho}$ and the vector of joint variables $\mathbf{q}$,

$$\mathbf{f} = \mathbf{f}(\boldsymbol{\rho}, \mathbf{q}) \tag{5.110}$$

For instance, in the DH model, $[\boldsymbol{\rho}^T, \mathbf{q}^T]^T = [\boldsymbol{\alpha}^T, \mathbf{a}^T, \mathbf{d}^T, \boldsymbol{\theta}^T]^T$. For an all-revolute robot, $\boldsymbol{\rho} = [\boldsymbol{\alpha}^T, \mathbf{a}^T, \mathbf{d}^T]^T$ while $\mathbf{q} = \boldsymbol{\theta}$, where $\boldsymbol{\alpha}$ is the vector of all link twist angles, $\mathbf{a}$ is the vector of all common normal lengths, $\mathbf{d}$ is the vector of offset distances between consecutive common normals, and $\boldsymbol{\theta}$ is the vector of joint rotations.

The kinematic errors of a robot are composed of two parts: nongeometric and geometric errors. Geometric errors can be modeled as link parameter deviations

$\Delta\boldsymbol{\rho}$ and joint offsets $\Delta\mathbf{q}_{\text{off}}$. Nongeometric errors due to effects such as joint compliance and link deflection are hard to model. For simplicity only a subclass of nongeometric errors, namely those that can be modeled as nonlinear functions of the joint commands are considered. We denote by $\Delta\mathbf{q}_{\text{ng}}$ the equivalent joint commands that produce the overall effect of nongeometric errors. The variables $\Delta\boldsymbol{\rho}$, $\Delta\mathbf{q}_{\text{off}}$, and $\Delta\mathbf{q}_{\text{ng}}$ are assumed available from the parameter identification phase. Denote

$$\Delta\mathbf{q} = \Delta\mathbf{q}_{\text{off}} + \Delta\mathbf{q}_{\text{ng}} \tag{5.111}$$

If no data are available concerning nongeometric errors, the algorithm assumes $\Delta\mathbf{q}_{\text{ng}} = \mathbf{0}$. The objective is to choose joint variables adjustment $\Delta\mathbf{q}_{\text{c}}$ to minimize $\|\Delta\mathbf{f}\|$ at every task point $\mathbf{q}^0$, where

$$\Delta\mathbf{f} = \mathbf{f}(\boldsymbol{\rho}^0 + \Delta\boldsymbol{\rho}, \mathbf{q}^0 + \Delta\mathbf{q} + \Delta\mathbf{q}_{\text{c}}) - \mathbf{f}(\boldsymbol{\rho}^0, \mathbf{q}^0) \tag{5.112}$$

Since the kinematic parameter errors are assumed small, a linear approximation is used to estimate $\Delta\mathbf{f}$ in terms of the identified kinematic parameter errors.

$$\Delta\mathbf{f} \simeq \left.\frac{\partial\mathbf{f}}{\partial\boldsymbol{\rho}}\right|_{\boldsymbol{\rho}^0, \mathbf{q}^0} \Delta\boldsymbol{\rho} + \left.\frac{\partial\mathbf{f}}{\partial\mathbf{q}}\right|_{\boldsymbol{\rho}^0, \mathbf{q}^0} \Delta\mathbf{q} + \left.\frac{\partial\mathbf{f}}{\partial\mathbf{q}}\right|_{\boldsymbol{\rho}^0, \mathbf{q}^0} \Delta\mathbf{q}_{\text{c}} \tag{5.113}$$

where the matrix kinematic sensitivities $\partial\mathbf{f}/\partial\boldsymbol{\rho}$ and $\partial\mathbf{f}/\partial\mathbf{q}$ depend on the nominal model of the robot at the particular task point. Denote

$$\left.\frac{\partial\mathbf{f}}{\partial\boldsymbol{\rho}}\right|_{\boldsymbol{\rho}^0, \mathbf{q}^0} = \mathbf{A}_{\text{f}} \tag{5.114}$$

$$\left.\frac{\partial\mathbf{f}}{\partial\mathbf{q}}\right|_{\boldsymbol{\rho}^0, \mathbf{q}^0} = \mathbf{B}_{\text{f}} \tag{5.115}$$

then

$$\Delta\mathbf{f} \simeq \mathbf{A}_{\text{f}}\Delta\boldsymbol{\rho} + \mathbf{B}_{\text{f}}\Delta\mathbf{q} + \mathbf{B}_{\text{f}}\Delta\mathbf{q}_{\text{c}} \tag{5.116}$$

where $A_{\text{f}} \in R^{M \times 12}$ and $B_{\text{f}} \in R^{N \times 12}$. Here M is the number of link parameters in the whole model. For example, if the DH model is used, $M = 3 \times N$. Equation 5.113 will be referred to as a *redundant linearized error model* since the vector $\mathbf{f}$ is a redundant description of the robot kinematics, compared to a description using a 6-vector. Let

$$\mathbf{x}_{\text{c}} = \mathbf{A}_{\text{f}}\Delta\boldsymbol{\rho} + \mathbf{B}_{\text{f}}\Delta\mathbf{q} \tag{5.117}$$

$$\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_N] = \mathbf{B}_{\text{f}} \tag{5.118}$$

and

$$\mathbf{u} = [u_1, u_2, \ldots, u_N]^T = [\Delta q_{\text{c},1}, \Delta q_{\text{c},2}, \ldots, \Delta q_{\text{c},N}]^T \tag{5.119}$$

where $\mathbf{b}_k$, $\mathbf{x}_c \in \mathbf{R}^{12}$. Then Equation 5.113 can be rewritten as the following discrete-time state equation

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{b}_k u_k \qquad k = 1, 2, \ldots, N \tag{5.120}$$

$$\mathbf{x}_1 = \mathbf{x}_c \tag{5.121}$$

The problem now takes a form of the standard optimal control problem discussed in the previous section. A cost function J_x representing the end-point accuracy may be chosen as follows:

$$J_x = \mathbf{x}_{N+1}^T \mathbf{Q}_x \mathbf{x}_{N+1} \tag{5.122}$$

The solution can be found by using the least-squares method. However, to include the size of the control signal as part of the cost and by that also avoid singularity problems, a modified cost function is selected.

5.3.3 Optimal Control Solution to the Calibration Compensation Problem

The solution of the linear optimal control problem is the vector $\mathbf{u}$ of the joint variable corrections (or compensations) that minimizes the modified cost function $J(\mathbf{u})$

$$J(\mathbf{u}) = \mathbf{x}_{N+1}^T \mathbf{Q}_x \mathbf{x}_{N+1} + \sum_{k=1}^{N} \gamma_k u_k^2 \tag{5.123}$$

subject to Equations 5.120 and 5.121. The terminal error weighting matrix, $\mathbf{Q}_x$, is taken to be symmetric nonnegative definite, and the control weights γ_k are strictly positive.

The optimal solution u_k^* is given by the feedback control law (Equation 5.84) that is repeated here for clarity:

$$u_k^* = \mathbf{k}_k^T \mathbf{x}_k \tag{5.124}$$

where $\mathbf{k}_k^T$ is obtained from a recursive solution of a related matrix Riccati difference equation as shown in the previous section:

$$\mathbf{k}_{N-k}^T = -\frac{\mathbf{b}_{N-k}^T \mathbf{P}_k}{\mathbf{b}_{N-k}^T \mathbf{P}_k \mathbf{b}_{N-k} + \gamma_{N-k}}, \qquad k = 0, 1, \ldots, N-1 \tag{5.125}$$

$$\mathbf{P}_{k+1} = \mathbf{P}_k + \mathbf{P}_k \mathbf{b}_{N-k}^T \mathbf{b}_{N-k} \tag{5.126}$$

with

$$\mathbf{P}_0 = \mathbf{Q}_x \tag{5.127}$$

The resulting minimum cost is

$$J^* = \mathbf{x}_1^T \mathbf{P}_N \mathbf{x}_1 \tag{5.128}$$

Note that if $\gamma_k = 0$, for any $k = 0, \ldots, N-1$, then $\mathbf{k}_k^T$ becomes undefined at singularities of the nominal kinematic model. The choice of the modified cost function (Equation 5.123) ensures the existence and uniqueness of the optimal solution.

Later in this section a method of choosing $\mathbf{Q}_x$ and γ_k is presented. After $\Delta\mathbf{q}_c = \mathbf{u}$ is obtained, the actual vector of joint variable $\mathbf{q}_{act}$ to be used at this particular task point is

$$\mathbf{q}_{act} = \mathbf{q}^0 + \Delta\mathbf{q}_c \tag{5.129}$$

The procedure is now repeated at every task point.

Remarks:

1. The resulting joint command may in certain cases exceed the joint travel boundaries. The solution is no longer optimal if a saturated joint command is taken. To prevent this, one may iteratively readjust the weighting coefficients γ_k. In addition, safety margins at the joint boundaries must be left when the task is created, leaving room for calibration corrections. More discussion of the subject will be given later in this section.
2. The pose error after compensation in general cannot be made zero even if the identification phase of the calibration process is perfect. The optimal cost value according to Equation 5.88 is in general nonzero.
3. $\mathbf{x}_c$ is the vector of the linearized initial pose error and $\mathbf{x}_{N+1}$ is the vector of the linearized final pose error. The performance of the compensation may be determined from the ratio of J^* (given in Equation 5.88) and $\mathbf{x}_c^T \mathbf{Q}_x \mathbf{x}_c$.

The selection process of $\mathbf{Q}_x$ and γ_k is explained next. Assuming initially $\mathbf{Q}_x$ to be the identity matrix, the first term in the cost function (Equation 5.123) is then the square of the Euclidean norm of the pose error vector. The first nine entries of the $\Delta\mathbf{f}$ vector represent orientation errors and the other three elements represent positioning errors. The positioning errors are absolute errors having sometimes orders of magnitude difference in scaling with respect to the orientation errors depending on the selection of unit systems. The weighting matrix $\mathbf{Q}_x$ can instead be selected to relate the cost function to the relative errors. This may be achieved through choosing a diagonal $\mathbf{Q}_x$ in the following way. Let $\mathbf{Q}_x = \text{diag}[q_{11}, q_{22}, \ldots, q_{12,12}]$, where

$$q_{ii} = \frac{q'_{ii}}{\|\mathbf{n}\|^2} = q'_{ii}, \qquad i = 1, 2, 3 \tag{5.130}$$

$$q_{ii} = \frac{q'_{ii}}{\|\mathbf{o}\|^2} = q'_{ii} \qquad i = 4, 5, 6 \tag{5.131}$$

$$q_{ii} = \frac{q'_{ii}}{\|\mathbf{a}\|^2} = q'_{ii} \qquad i = 7, 8, 9 \tag{5.132}$$

$$q_{ii} = \frac{q'_{ii}}{\|\mathbf{p}\|^2} \qquad i = 10, 11, 12 \tag{5.133}$$

Thus

$$\mathbf{Q}_x = \text{diag}\left[q'_{11}, \ldots, q'_{99}, \frac{q'_{10,10}}{\|\mathbf{p}^2\|}, \frac{q'_{11,11}}{\|\mathbf{p}^2\|}, \frac{q'_{12,12}}{\|\mathbf{p}^2\|} \right] \tag{5.134}$$

The particular choice of values q'_{ii} now reflects different accuracy requirements of positioning and orientation of the end effector. For instance, let $q'_{ii} = k_1$ for $i = 1, \ldots, 9$, and $q'_{ii} = k_2$ for $i = 10, 11, 12$. Then

$$\mathbf{Q}_x = \text{diag}\left[k_1, \ldots, k_1, \frac{k_2}{\|\mathbf{p}\|^2}, \frac{k_2}{\|\mathbf{p}\|^2}, \frac{k_2}{\|\mathbf{p}\|^2} \right] \tag{5.135}$$

The values of γ_k may be chosen to achieve several objectives. First, the correction $\Delta\mathbf{q}_c$ also needs to be normalized whenever the robot features both types of joints, revolute and prismatic, with different scalings. Second, at task points that require one or more of the joint variables to be near the joint travel boundaries or near robot singularities, large values of γ_k need to be selected to reduce the amount of correction. A suggested method of choosing γ_k is as follows:

$$\gamma_k = \delta_{\gamma,k} \frac{\gamma}{q_{k,\text{rang}}}, \qquad \gamma > 0 \tag{5.136}$$

where $q_{k,\text{rang}}$, the square of the total kth joint travel, is used to normalize the corrections in the case of the robot with two types of joints. In the case of an all-revolute manipulator one may set $q_{k,\text{rang}} = 1$.

Acting as a penalty coefficient, $\delta_{\gamma,k}$ may be chosen to be inversely proportional to the minimum distance between the kth nominal joint command and its boundaries, or it could be chosen as a switching function, being a very large number only when the above distance is smaller than a preset threshold value. At the limit, $\delta_{\gamma,k} = \infty$, which results in a zero correction. Large values $\delta_{\gamma,k}$ may degrade the correction as the cost function of Equation 5.123 is driven away from that of Equation 5.122. Again an alternative is that safety margins are maintained during the programming of the application. If such margins are sufficiently large, $\delta_{\gamma,k}$ may be set to one.

Coming back to the linearized accuracy error $\Delta\mathbf{f}$ of Equation 5.113, we notice that the linearization is done with respect to both the joint variables as well as the fixed kinematic parameters. Actually it is necessary to linearize only with respect to the joint variables since $\boldsymbol{\rho} = \boldsymbol{\rho}^0 + \Delta\boldsymbol{\rho}$ and $\mathbf{q} = \mathbf{q}^0 + \Delta\mathbf{q}$ are assumed known following the kinematic parameter identification. Let $\Delta\mathbf{f}^{(m)}$ denote a modified expression for the accuracy error vector as follows:

$$\Delta \mathbf{f}^{(m)} = \mathbf{f}(\boldsymbol{\rho}, \mathbf{q}) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{q}}\right|_{\rho, q} \Delta \mathbf{q}_c - f(\boldsymbol{\rho}^0, \mathbf{q}^0) \tag{5.137}$$

where $\partial \mathbf{f}/\partial \mathbf{q}$ depends on the actual parameters $\boldsymbol{\rho}$ and $\mathbf{q}$. Assuming that $\Delta \boldsymbol{\rho}$ and $\Delta \mathbf{q}$ are small, one can approximate $\partial \mathbf{f}/\partial \mathbf{q}|_{\rho, q}$ as follows:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\boldsymbol{\rho}, \mathbf{q}) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{q}}\right|_{\rho^0, q^0} + \left.\frac{\partial^2 \mathbf{f}}{\partial \mathbf{q} \partial \boldsymbol{\rho}}\right|_{\rho^0, q^0} \Delta \boldsymbol{\rho} + \left.\frac{\partial^2 \mathbf{f}}{\partial \mathbf{q} \partial \boldsymbol{\rho}}\right|_{\rho^0, q^0} \Delta \mathbf{q} \tag{5.138}$$

Substitution of Equation 5.138 into Equation 5.137 and ignoring higher order terms of $\Delta \mathbf{q}$ and $\Delta \boldsymbol{\rho}$ yields

$$\Delta \mathbf{f}^{(m)} = \mathbf{f}(\boldsymbol{\rho}, \mathbf{q}) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{q}}\right|_{\rho^0, q^0} \Delta \mathbf{q}_c^{(m)} - \mathbf{f}(\boldsymbol{\rho}^0, \mathbf{q}^0) \tag{5.139}$$

The same optimal control solution method can now be applied to find $\Delta \mathbf{q}_c^{(m)}$ by defining a new initial vector $\mathbf{x}_c^{(m)}$:

$$\mathbf{x}_c^{(m)} = \mathbf{f}(\boldsymbol{\rho}, \mathbf{q}) - \mathbf{f}(\boldsymbol{\rho}^0, \mathbf{q}^0) \tag{5.140}$$

Remark: Expressions 5.139–5.140 provide also a conceptual solution to the *automated robot reprogramming* problem. The reprogramming is done by using endpoint sensing to determine the end effector location without ever identifying the actual kinematic model. Let $\mathbf{q}^0$ be the programmed joint commands vector, $\mathbf{f}(\boldsymbol{\rho}^0, \mathbf{q}^0)$ be the position of the desired task point and $\mathbf{x}_{ac}(\mathbf{q}^0)$ be the actual position of the robot's end effector when the command $\mathbf{q}^0$ is applied. $\mathbf{x}_{ac}(\mathbf{q}^0)$ is measurable using the same type of measurement techniques that are described in Chapter 3. Define

$$\mathbf{x}_c^{(rp)} = \mathbf{x}_{ac}(\mathbf{q}^0) - \mathbf{f}(\boldsymbol{\rho}^0, \mathbf{q}^0) \tag{5.141}$$

where $\mathbf{x}_c^{(rp)}$ denotes an initial condition for an optimal control algorithm that calculates the joint command increment $\Delta \mathbf{q}_c^{(rp)}$. Mathematically, $\Delta \mathbf{q}_c^{(rp)}$ and $\Delta \mathbf{q}_c^{(m)}$ are completely identical. The only difference is that $\mathbf{x}_c^{(m)}$ is a calculated entity while $\mathbf{x}_c^{(rp)}$ is a measured entity.

The modified calibration compensation solution based on Equations 5.139 through 5.140 becomes particularly attractive in cases where large kinematic parameter discontinuities exist as in the case of using a DH modeling convention for a robot that has nominally parallel consecutive joint axes. In such cases, Equation 5.113 is no longer valid while Equation 5.139 is.

The only advantage of the fully linearized redundant model Equation 5.113 over the partially linearized redundant model of Equation 5.139 is that Equation 5.113 is in the form that allows model reduction to a nonredundant error model along the lines to be discussed next.

5.3.4 Calibration for Reduced-Order Error Models

The computation effort can be drastically reduced through simplification of the redundant error models presented in the previous section. In this section an equivalent reduced-order error model is derived followed by a computational cost comparison between the redundant and simplified error models.

Considering the homogeneous transformation matrix $\mathbf{T}_N$ of Equation 5.109, let $d\mathbf{T}_N$ be a differential transformation as described by Paul [10]:

$$d\mathbf{T}_N = \mathbf{T}_N \cdot \delta\mathbf{T}_N \tag{5.142}$$

where

$$\delta\mathbf{T}_N = \begin{bmatrix} 0 & -\delta z_N & \delta y_N & dx_N \\ \delta z_N & 0 & -\delta x_N & dy_N \\ -\delta y_N & \delta x_N & 0 & dz_N \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.143}$$

$[dx_N, dy_N, dz_N]^T$ are the translational errors and $[\delta x_N, \delta y_N, \delta z_N]^T$ are the rotational errors of the end effector. $d\mathbf{T}_N$ is a known matrix. It can be obtained from $\mathbf{T}_N$ in one of the two following methods:

1. Linearization with respect to $\boldsymbol{\rho}$ and $\mathbf{q}$, as follows:

$$d\mathbf{T}_N = \left.\frac{\partial \mathbf{T}_N}{\partial \boldsymbol{\rho}}\right|_{\boldsymbol{\rho}^0, \mathbf{q}^0} \Delta\boldsymbol{\rho} + \left.\frac{\partial \mathbf{T}_N}{\partial \mathbf{q}}\right|_{\boldsymbol{\rho}^0, \mathbf{q}^0} \Delta\mathbf{q} \tag{5.144}$$

or

2. Exact difference as follows:

$$d\mathbf{T}_N = \mathbf{T}_N(\boldsymbol{\rho}, \mathbf{q}) - \mathbf{T}_N(\boldsymbol{\rho}^0, \mathbf{q}^0) \tag{5.145}$$

where $\boldsymbol{\rho} = \boldsymbol{\rho}^0 + \Delta\boldsymbol{\rho}$ and $\mathbf{q} = \mathbf{q}^0 + \Delta\mathbf{q}$.

From Equation 5.142 one can now find $\delta\mathbf{T}_N$:

$$\delta\mathbf{T}_N = \mathbf{T}_N^{-1} \cdot \mathbf{dT}_N \tag{5.146}$$

The reader is referred to a similar procedure carried in Section 5.2.

Define a reduced-order accuracy error vector $\Delta\mathbf{g}$ as

$$\Delta\mathbf{g} = \mathbf{A}_g \Delta\boldsymbol{\rho} + \mathbf{B}_g \Delta\mathbf{q} + \mathbf{B}_g \Delta\mathbf{q}_c \tag{5.147}$$

where $\mathbf{A}_g \in R^{M\times 6}$ and $\mathbf{B}_g \in R^{M\times 6}$. The above equation will be referred to as a

reduced-order linearized error model. Similar to the derivation in the previous section let

$$\mathbf{y}_c = \mathbf{A}_g \Delta\boldsymbol{\rho} + \mathbf{B}_g \Delta\mathbf{q} \tag{5.148}$$

$$[\mathbf{b}_{1g}, \ldots, \mathbf{b}_{Ng}] = \mathbf{B}_g \tag{5.149}$$

$$\mathbf{u} = \Delta\mathbf{q}_c \tag{5.150}$$

where $\mathbf{b}_{ig}$, $\mathbf{y}_c \in R^6$ and $\mathbf{u} \in R^N$.

The vector $\mathbf{y}_c$ can be constructed from $\delta\mathbf{T}_N$ as $\mathbf{y}_c = [dx_N, dy_N, dz_N, \delta x_N, \delta y_N, \delta z_N]^T$. The matrix $\mathbf{B}_g$ is constructed as follows. Define a matrix $\mathbf{L}_k$

$$\mathbf{L}_k = \mathbf{T}_N^{-1} \frac{\partial \mathbf{T}_N}{\partial q_k}\bigg|_{\boldsymbol{\rho}^0, \mathbf{q}^0}, \qquad k = 1, \ldots, N \tag{5.151}$$

Then the vector $\mathbf{b}_{kg}$ of Equation 5.149 is the kth column of the matrix $\mathbf{B}_g$, where

$$[\mathbf{B}_g]_{i,k} = [\mathbf{L}_k]_{i,4} \qquad i = 1, 2, 3 \tag{5.152}$$

$$[\mathbf{B}_g]_{4,k} = [\mathbf{L}_k]_{3,2} \tag{5.153}$$

$$[\mathbf{B}_g]_{5,k} = [\mathbf{L}_k]_{1,3} \tag{5.154}$$

$$[\mathbf{B}_g]_{6,k} = [\mathbf{L}_k]_{2,1} \tag{5.155}$$

for $k = 1, \ldots, N$.

Equations 5.147 to 5.150 can now be rewritten as the following state-variable model:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \mathbf{b}_{kg} u_k, \qquad k = 1, 2, \ldots, N \tag{5.156}$$

$$\mathbf{y}_1 = \mathbf{y}_c \tag{5.157}$$

Minimizing the cost function $J(\mathbf{u})$, where

$$J(\mathbf{u}) = \mathbf{y}_{N+1}^T \mathbf{Q}_y \mathbf{y}_{N+1} + \sum_{k=1}^{N} \gamma_k u_k^2 \tag{5.158}$$

subject to the difference Equations 5.156 and 5.157. $\mathbf{Q}_y$ is given in Equation 5.162 below and γ_k is positive for every $k \in [1, \ldots, N]$.

The optimal solution of u_k is given by the feedback control law

$$u_k^* = \mathbf{k}_{kg}^T \mathbf{y}_k \tag{5.159}$$

where $\mathbf{k}_{kg}^T$ is obtained from a recursive solution similar to Equation 5.85 with the initial condition $\mathbf{P}_0 = \mathbf{Q}_y$.

The key problem is to construct a cost function J_y so that minimizing J_y is equivalent to minimizing J_x (of Equation 5.122), where

$$J_y = \mathbf{y}_{N+1}^T \mathbf{Q}_y y_{N+1} \tag{5.160}$$

It will be shown now that J_y is equal to J_x if the matrices $\mathbf{Q}_x$ and $\mathbf{Q}_y$ satisfy the conditions given below

Proposition: Let

$$\mathbf{Q}_x = \text{diag}[q_{11}, q_{22}, \ldots, q_{12,12}] \tag{5.161}$$

then $J_x = J_y$ if

$$\mathbf{Q}_y = \begin{bmatrix} \mathbf{Q}_{3\times3}^u & \mathbf{O}_{3\times3} \\ \mathbf{O}_{3\times3} & \mathbf{Q}_{3\times3}^l \end{bmatrix} \tag{5.162}$$

where the upper left block $\mathbf{Q}_{3\times3}^u$ is

$$\mathbf{Q}_{3\times3}^u = \begin{bmatrix} \mathbf{n}^T\mathbf{Q}_{10-12}\mathbf{n} & \mathbf{n}^T\mathbf{Q}_{10-12}\mathbf{o} & \mathbf{n}^T\mathbf{Q}_{10-12}\mathbf{a} \\ \mathbf{n}^T\mathbf{Q}_{10-12}\mathbf{o} & \mathbf{o}^T\mathbf{Q}_{10-12}\mathbf{o} & \mathbf{o}^T\mathbf{Q}_{10-12}\mathbf{a} \\ \mathbf{n}^T\mathbf{Q}_{10-12}\mathbf{a} & \mathbf{o}^T\mathbf{Q}_{10-12}\mathbf{a} & \mathbf{a}^T\mathbf{Q}_{10-12}\mathbf{a} \end{bmatrix} \tag{5.163}$$

and the lower right block $\mathbf{Q}_{3\times3}^l$ is

$$\mathbf{Q}_{3\times3}^l = \begin{bmatrix} \mathbf{o}^T\mathbf{Q}_{4-6}\mathbf{o} + \mathbf{a}^T\mathbf{Q}_{7-9}\mathbf{a} & -\mathbf{n}^T\mathbf{Q}_{7-9}\mathbf{o} & -\mathbf{n}^T\mathbf{Q}_{4-6}\mathbf{a} \\ -\mathbf{n}^T\mathbf{Q}_{7-9}\mathbf{o} & \mathbf{a}^T\mathbf{Q}_{7-9}\mathbf{a} + \mathbf{n}^T\mathbf{Q}_{1-3}\mathbf{n} & -\mathbf{o}^T\mathbf{Q}_{1-3}\mathbf{a} \\ -\mathbf{n}^T\mathbf{Q}_{4-6}\mathbf{a} & -\mathbf{o}^T\mathbf{Q}_{1-3}\mathbf{a} & \mathbf{n}^T\mathbf{Q}_{1-3}\mathbf{n} + \mathbf{o}^T\mathbf{Q}_{4-6}\mathbf{o} \end{bmatrix} \tag{5.164}$$

where $\mathbf{n}$, $\mathbf{o}$, and $\mathbf{a}$ are defined in Equation 5.109, and $\mathbf{Q}_{1-3}$, $\mathbf{Q}_{4-6}$, $\mathbf{Q}_{7-9}$, and $\mathbf{Q}_{10-12}$ are 3×3 diagonal matrices as follows:

$$Q_{1-3} = \text{diag}[q_{11}, q_{22}, q_{23}] \tag{5.165}$$

$$Q_{4-6} = \text{diag}[q_{44}, q_{55}, q_{66}] \tag{5.166}$$

$$Q_{7-9} = \text{diag}[q_{77}, q_{88}, q_{99}] \tag{5.167}$$

$$Q_{10-12} = \text{diag}[q_{10,10}, q_{11,11}, q_{12,12}] \tag{5.168}$$

Specifically if (as in Equations 5.135)

$$\mathbf{Q}_x = \text{diag}\left[k_1, \ldots, k_1, \frac{k_2}{\|\mathbf{p}\|^2}, \frac{k_2}{\|\mathbf{p}\|^2}, \frac{k_2}{\|\mathbf{p}\|^2}\right] \tag{5.169}$$

then Equations 5.162 takes the simple form of

$$\mathbf{Q}_y = \text{diag}\left[\frac{k_2}{\|\mathbf{p}\|^2}, \frac{k_2}{\|\mathbf{p}\|^2}, \frac{k_2}{\|\mathbf{p}\|^2}, 2k_1, 2k_1, 2k_1\right] \tag{5.170}$$

Proof: Denote

$$d\mathbf{T}_N = \begin{bmatrix} dt_{11} & dt_{12} & dt_{13} & dt_{14} \\ dt_{21} & dt_{22} & dt_{23} & dt_{24} \\ dt_{31} & dt_{32} & dt_{33} & dt_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.171}$$

By Equation 5.109 and the definition of $\mathbf{f}$, $\mathbf{x}_c = [dt_{11}, dt_{21}, \ldots, dt_{34}]^T$. By definition of Δg, $\mathbf{y}_c = [dx_N, dy_N, dz_N, \delta x_N, \delta y_N, \delta z_N]^T$. Thus by Equation 5.142, the elements of the vector $\mathbf{x}_c$ can be expressed in terms of the elements of $\mathbf{y}_c$ as follows:

$$\begin{aligned}
x_{c,1} &= o_x y_{c,6} - a_x y_{c,5} \\
x_{c,2} &= o_y y_{c,6} - a_y y_{c,5} \\
x_{c,3} &= o_z y_{c,6} - a_z y_{c,5} \\
x_{c,4} &= -n_x y_{c,6} + a_x y_{c,4} \\
x_{c,5} &= -n_y y_{c,6} + a_y y_{c,4} \\
x_{c,6} &= -n_z y_{c,6} + a_z y_{c,4} \\
x_{c,7} &= n_x y_{c,5} - o_x y_{c,4} \\
x_{c,8} &= n_y y_{c,5} - o_y y_{c,4} \\
x_{c,9} &= n_z y_{c,5} - o_z y_{c,4} \\
x_{c,10} &= n_x y_{c,1} + o_x y_{c,2} + a_x y_{c,3} \\
x_{c,11} &= n_y y_{c,1} + o_y y_{c,2} + a_x y_{c,3} \\
x_{c,12} &= n_z y_{c,1} + o_z y_{c,2} + a_z y_{c,3}
\end{aligned}$$

$J_x(\mathbf{x}_c)$ is a quadratic form of $\mathbf{y}_c$ as shown by the following straightforward derivation:

$$\begin{aligned}
J_x(\mathbf{x}_c) = \mathbf{x}_c \mathbf{Q}_x \mathbf{x}_c = {} & (\mathbf{n}^T \mathbf{Q}_{10-12} \mathbf{n}) y_{c,1}^2 + (\mathbf{o}^T \mathbf{Q}_{10-12} \mathbf{o}) y_{c,2}^2 \\
& + (\mathbf{a}^T \mathbf{Q}_{10-12} \mathbf{a}) y_{c,3}^2 + 2(\mathbf{n}^T \mathbf{Q}_{10-12} \mathbf{o}) y_{c,1} y_{c,2} \\
& + 2(\mathbf{n}^T \mathbf{Q}_{10-12} \mathbf{a}) y_{c,1} y_{c,3} + 2(\mathbf{o}^T \mathbf{Q}_{10-12} \mathbf{a}) y_{c,2} y_{c,3} \\
& + (\mathbf{a}^T \mathbf{Q}_{7-9} \mathbf{a} + \mathbf{o}^T \mathbf{Q}_{4-6} \mathbf{o}) y_{c,4}^2 \\
& + (\mathbf{a}^T \mathbf{Q}_{7-9} \mathbf{a} + \mathbf{n}^T \mathbf{Q}_{1-3} \mathbf{n}) y_{c,5}^2 \\
& + (\mathbf{o}^T \mathbf{Q}_{4-6} \mathbf{o} + \mathbf{n}^T \mathbf{Q}_{1-3} \mathbf{n}) y_{c,6}^2 \\
& - 2(\mathbf{o}^T \mathbf{Q}_{7-9} \mathbf{o}) y_{c,4} y_{c,5} - 2(\mathbf{n}^T \mathbf{Q}_{4-6} \mathbf{a}) y_{c,4} y_{c,6} \\
& - 2(\mathbf{o}^T \mathbf{Q}_{1-3} \mathbf{a}) y_{c,5} y_{c,6}
\end{aligned}$$

Since $\mathbf{y}_c$ is arbitrary, equating the above to $\mathbf{y}_c^T \mathbf{Q}_y \mathbf{y}_c$ results in $\mathbf{Q}_y$ being in the form of Equation 5.162. Since $\mathbf{Q}_x$ is nonnegative definite and the equality holds for any $(\mathbf{x}_c, \mathbf{y}_c)$, then the resulting $\mathbf{Q}_y$ is also nonnegative definite.

By Equation 5.135 and the orthogonality of $\mathbf{n}$, $\mathbf{o}$, $\mathbf{a}$, Equation 5.170 results.

□

Further simplifications to the optimal control algorithm are possible for manipulators with particular geometry. For example, if the manipulator is 3T-3R, then the state equation given in Equation 5.156 decouples since the orientation error vector $\boldsymbol{\delta} = [\delta x, \delta y, \delta z]$ is not a function of the prismatic joint corrections $\Delta \mathbf{d}$. The problem can now be solved using two consecutive linear quadratic regulator algorithms, where the first one is for correction of the orientation deviation $\boldsymbol{\delta}$, and the second one is for correcting the position errors without changing the orientation vector that has been corrected by the first algorithm.

Two factors contribute to the computational complexity of the optimal control compensation strategy:

1. Computation of the coefficient matrix $\mathbf{B}_f$ (or $\mathbf{B}_g$) and the initial conditions $\mathbf{x}_c$ (or $\mathbf{y}_c$) of the state equations.
2. Computation of the correction vector $\Delta \mathbf{q}_c$ after $\mathbf{B}_f$ (or $\mathbf{B}_g$) and $\mathbf{x}_c$ (or $\mathbf{y}_c$) are made available. The complexity of $\mathbf{B}_f$ (or $\mathbf{B}_g$) depends on the choice of the kinematic model.

To compute $\mathbf{k}_{N-k}^T$, about $n(n+1)(M+A)$ operations are needed, where M stands for multiplication, A stands for addition, and n is the dimension of the weighting matrix $\mathbf{Q}_x$ (or $\mathbf{Q}_y$). There are N steps, therefore $Nn(n+1)(M+A)$ operations are needed. However if $\mathbf{Q}_x$ (or $\mathbf{Q}_y$) is diagonal, the number of operations reduces to $(N-1)n(n+1)(M+A) + 2n(M+A)$. To compute $\mathbf{P}_k$, $n(n+1)(M+A) + n^2A$ operations are needed. $N[n(n+1)(M+A) + n^2A]$ operations are needed for N steps. $\mathbf{P}_N$, however, is needed only for the calculation of the minimum cost J^*. Therefore, if J^* is not explicitly required, the number of operations is reduced to $(N-1)[n(n+1)(M+A) + n^2A]$. Again for a diagonal $\mathbf{Q}_x(\mathbf{Q}_y)$ matrix, the computation cost for P_1 through P_{N-1} requires $(N-2)[n(n+1)(M+A) + n^2A]$ operations.

After $\mathbf{b}_k$ becomes available, computation of $\mathbf{u}_k$ for N steps needs about $Nn(M+A)$ operations.

Assume that the minimum cost J^* needs not be calculated, then the total number of operations T_{op} for computing the compensation algorithm given $\mathbf{B}_f(\mathbf{B}_g)$ and $\mathbf{x}_c(\mathbf{y}_c)$ is approximately:

$$T_{op} = (N-1)[2n(n+1)(M+A) + n^2A] + n(n+1)(M+A) + n^2A + 2Nn(M+A) \tag{5.172}$$

If Q_x (or Q_y) is diagonal, then

$$T_{op} = (N-2)[2n(n+1)(M+A) + n^2A] + n(n+5)(M+A) + n^2A + 2Nn(M+A) \tag{5.173}$$

For the redundant error model $n = 12$ and by taking a diagonal matrix Q_x the resulting number of operations is $T_{op} = 1596M + 2316A$. For the reduced order error model $n = 6$ and for Q_y being diagonal, $T_{op} = 474M + 654A$. Using a reduced order model requires about one-quarter of the computational effort required by the corresponding redundant error model after $\mathbf{x}_c$ (or $\mathbf{y}_c$) and $\mathbf{B}_f$ (or $\mathbf{B}_g$) are available.

For a 3T-3R type of robot, computing each correction vector needs only about 290 multiplications and 390 additions after the initial conditions and the coefficients of the state equations are set.

5.3.5 Simulation Example of PUMA 560 Robot

The kinematic model of the Unimation PUMA 560 robot as described in [5] was adopted for this simulation. The kinematic parameters using DH convention are listed in Table 5.3. The simulated kinematic errors are listed in Table 5.4.

Both the reduced-order and redundant linearized error models were used to obtain the required joint variable corrections $\Delta\boldsymbol{\theta}$. Simulations show that the outputs of the two algorithms match closely. The linear quadratic regulator algorithm using the reduced error model is, of course, much faster. In Table 5.5, results obtained by the linear quadratic regulator algorithm based on the reduced-order error model are given to illustrate the effectiveness of the algorithms presented in this section.

TABLE 5.3. Link Parameters of the PUMA 560

Link Number	α_i (deg)	a_i (mm)	d_i (mm)	θ_i (deg)
1	90	0	0	$-160 \rightarrow 160$
2	0	431.8	149.09	$-225 \rightarrow 45$
3	90	−20.32	0	$-45 \rightarrow 225$
4	0	0	433.07	$-110 \rightarrow 170$
5	0	0	0	$-100 \rightarrow 100$
6	0	0	56.25	$-266 \rightarrow 266$

TABLE 5.4. Simulated Parameter Deviations

Link Number	$\Delta\alpha_i$ (deg)	Δa_i (mm)	Δd_i (mm)	$\Delta\theta_i$ (deg)
1	0.1	0.05	0.1	−0.05
2	−0.05	0.1	−0.075	−0.1
3	0.05	−0.1	0.05	−0.1
4	−0.15	0	0	−0.1
5	0.15	0	0	0.1
6	−0.13	0	0	0.1

TABLE 5.5. Simulation Results for $k_1 = k_2 = 1$ at a Nonsingular Robot Configuration

Error Norm	$\|\Delta \mathbf{n}\|$	$\|\Delta \mathbf{o}\|$	$\|\Delta \mathbf{a}\|$	$\|\Delta \mathbf{p}\|/\|\mathbf{p}\|$
Before correction	0.004990	0.004399	0.002369	0.000796
After correction*	0.000010	0.000006	0.000011	0.000008
After correction**	0.000010	0.000002	0.000010	0.000007
After correction***	0.000011	0.000002	0.000011	0.000002

*$\gamma = 0.01$, $d\mathbf{T}_N$ is computed by Equation 5.144.
**$\gamma = 0.01$, $d\mathbf{T}_N$ is computed by Equation 5.145.
***$\gamma = 0.0001$, $d\mathbf{T}_N$ is computed by Equation 5.144.

Joint Variable Correction	$\Delta\theta_1$	$\Delta\theta_2$	$\Delta\theta_3$	$\Delta\theta_4$	$\Delta\theta_5$	$\Delta\theta_6$
For correction*	−0.038	0.116	−0.165	−0.013	−0.119	0.318
For correction**	−0.038	0.116	−0.165	−0.013	−0.120	0.318
For correction***	−0.038	0.117	−0.167	−0.014	−0.119	0.319

Note: $\Delta\theta_k$ is in degrees.

Table 5.5 illustrates the following facts: (1) The correction results are insensitive to the choice of computation methods of $d\mathbf{T}_N$. (2) The results are also insensitive to the choice of γ_k as long as the robot is not near a singular configuration. The nominal joint vector in this example is assumed to be $\boldsymbol{\theta}^0 = [90, -90, 45, 45, 45, 0]$. Line (*) lists the norms of the orientation and positioning errors after corrections when $d\mathbf{T}_N$ is computed by the linearized error model in Equation 5.144. Line (**) lists the results when $d\mathbf{T}_N$ is computed by Equation 5.145. In both cases, γ is the same. Line (***) gives the results when γ is chosen to be 0.0001 and $d\mathbf{T}_N$ is computed by the linearized model. Table 5.5 also lists the joint variables corrections corresponding to the various cases.

Table 5.6 shows that k_1 and k_2 (from Equation 5.135) can be used to adjust

TABLE 5.6. Simulation Results for $\gamma = 0.01$

Error Norm	$\|\Delta \mathbf{n}\|$	$\|\Delta \mathbf{o}\|$	$\|\Delta \mathbf{a}\|$	$\|\Delta \mathbf{p}\|/\|\mathbf{p}\|$
Before correction	0.004990	0.004399	0.002369	0.000796
After correction*	0.000010	0.000002	0.000010	0.000007
After correction**	0.005634	0.005139	0.002323	0.000008
After correction***	0.000007	0.000007	0.000010	0.001776

*$k_1 = k_2 = 1$.
**$k_1 = 0, k_2 = 1$.
***$k_1 = 1, k_2 = 0$.

Joint Variable Correction	$\Delta\theta_1$	$\Delta\theta_2$	$\Delta\theta_3$	$\Delta\theta_4$	$\Delta\theta_5$	$\Delta\theta_6$
For correction*	−0.038	0.116	−0.165	−0.013	−0.120	0.318
For correction**	−0.025	0.103	−0.148	−0.002	0.016	0.000
For correction***	0.116	−0.012	−0.012	−0.027	−0.060	0.184

Note: $\Delta\theta_k$ is in degrees.

TABLE 5.7. Simulation Results for $k_1 = k_2 = 1$ in a Singular Robot Configuration

Error Norm	$\|\Delta n\|$	$\|\Delta o\|$	$\|\Delta a\|$	$\|\Delta p\|/\|p\|$
Before correction	0.003367	0.004823	0.004809	0.000951
After correction*	0.000009	0.000395	0.000395	0.001807
After correction**	0.000619	0.000026	0.000619	0.000197
After correction***	0.000753	0.000024	0.000752	0.000051

*$\gamma = 0.01$.
**$\gamma = 0.0001$.
***$\gamma = 0.0$.

Joint Variable Correction	$\Delta\theta_1$	$\Delta\theta_2$	$\Delta\theta_3$	$\Delta\theta_4$	$\Delta\theta_5$	$\Delta\theta_6$
For correction*	0.128	0.024	0.152	1.465	−0.195	−1.415
For correction**	−0.021	0.107	−0.133	15.28	−0.127	−15.13
For correction**	0.038	0.017	−0.167	16.91	−0.119	−16.74

Note: $\Delta\theta_k$ is in degrees.

the results in tradeoff of positioning or orientation accuracy. Here the nominal joint variables are the same as those of the last example. The upper part of Table 5.6 lists the orientation and relative positioning errors before and after corrections. The lower part of Table 5.6 gives the corresponding joint variable corrections. $d\mathbf{T}_N$ is computed here by Equation 5.144.

Table 5.7 shows that γ_k plays a very important role when the robot is near a singular configuration. The nominal joint variable $\boldsymbol{\theta}^0$ is chosen as $[90, -90, 45, 45, -1, 0]$. Since $\theta_5 = -1.0^0$, the PUMA arm is near a wrist singularity. Simulation results show that the joint variable corrections will be very large if γ is set to or near zero, though the error norm is minimized. Here again $d\mathbf{T}_N$ is computed by Equation 5.144.

Obviously the degree of success in improving the accuracy of a manipulator depends strongly on the accuracy of identifying the kinematic errors of the manipulator. Another limitation of the method is due to the "small perturbation assumption," based on which linearized error models are derived. If the kinematic errors are too large, iterative methods may have to be employed.

5.4 NONPARAMETRIC ACCURACY COMPENSATION

The main literature sources on which this section is based are the works by Whitney and Shamma [11, 12]. A good short review can also be found in the chapter by Hollerbach [7].

Parametric models for some of the robot accuracy error sources, including all nongeometric factors, may be quite difficult to obtain. Some error factors are too difficult to be expressed analytically. Others may display local variations of a random nature. Although parametric models are convenient because they are global and cover the whole workspace, a failure to model an error source may cause model-based methods to lose accuracy significantly. One may need, in

certain cases, to counter model-based identification and compensation with methods that are more empirical, use abstract interpolation functions, and may be valid only locally. Such an empirical approach, on the other hand, is independent of the need to model every error source. The idea is to approach the robot as a "black box," or more exactly, to approach the accuracy error as a "black box." One may still want to benefit from the robot nominal kinematics as a prime source of useful information.

The objective is to find a set of approximating functions that relate statically (or, in other words, algebraically), a given set of input data to a given set of measured output data. The measurement phase for nonparametric calibration is not quite the same as with the model-based method. First, the robot is moved through a set of configurations. At each configuration, the joint position encoders are read (this is the *input data*) and the end effector position and orientation in world coordinates is measured using end-point sensing. The difference between the measured pose at the measurement test points and the estimated pose when using the nominal model yields the accuracy error at each test point. This is normally the output data for model-based calibration. For nonparametric calibration, however, one additional measurement step is required. The pose error at each test point is manually compensated and the necessary amount of joint displacement correction is recorded. The approximating function that relates algebraically, at a given test point, the vector of joint transducer readings to the vector of joint displacement corrections usually has no direct physical significance. It is purely an empirical curve fit model. One attempts to find a function that will "best" account for the joint corrections in all tests points, given the joint readings in these test points. This "best" approximating function of the joint readings can then be used to compute a precise inverse kinematics solution (i.e., calculation of joint commands for desired arbitrary robot poses). The "black box" approach to robot calibration is illustrated in Figure 5.3.

Among the many available classes of approximating functions, two have been the subject of active research in robot calibration. One is based on multivariate polynomials to be discussed in Section 5.4.1 and the other is based upon table lookup schemes such as CMAC, discussed in Section 5.4.2.

Calibration that is based on a nonparametric approach applies best to robot accuracy enhancement only in the regions of the workspace where data were taken on which the coefficients of the approximation functions were based. Collecting calibration data over the entire robot workspace is in most cases not feasible. In the work by Whitney and Shamma [11, 12] it was reported that a robot with one set of approximating functions could be accurately calibrated over about a quarter of the robot's workspace.

5.4.1 Robot Calibration Using Polynomial Approximating Functions

The basic ideas of using polynomial approximating functions to relate the joint encoder readings to accuracy errors can be described as follows [11, 12]:

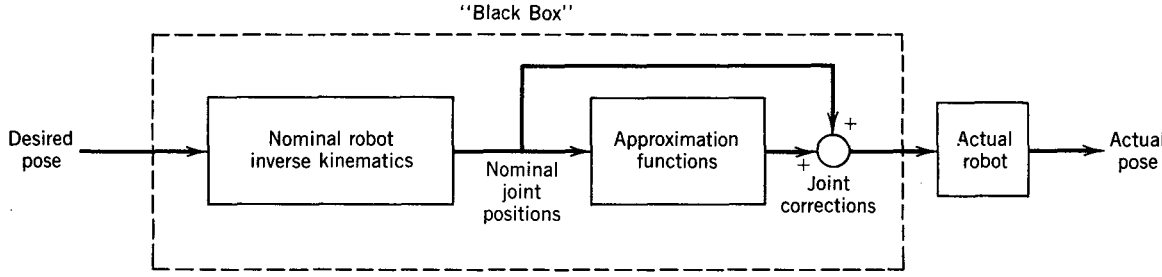


Figure 5.3. Nonparametric robot calibration.

Let $\boldsymbol{\eta}_i$ be the vector of joint encoder readings at the ith measurement point, $i = 1, \ldots, m$. Let $\Delta\boldsymbol{\eta}_i$ be the measured joint compensation at the same test point i. We denote by $(\Delta\eta_k)_i$ the kth joint correction, $k = 1, \ldots, K$. There will be two cases:

1. $K = 3$. This case relates to a study of a 3 DOF manipulator in which case $\boldsymbol{\eta}_i$ is the position error vector.
2. $K = 6$. The study of a general manipulator in which $\boldsymbol{\eta}_i$ is the 6-vector of position and orientation errors.

As was shown [11, 12], the solution of the first case plays an important role in the solution procedure of the second more general case.

Let $\beta_j(\boldsymbol{\eta})$ be the jth multivariate polynomial, $j = 0, \ldots, L-1$ in the elements η_l of the joint encoder readings vector $\boldsymbol{\eta}$, $l = 1, \ldots, N$, where N is the number of manipulator's degrees of freedom. For example, such a polynomial of order 2 takes the general form of

$$\beta_j(\boldsymbol{\eta}) = \alpha_0^{(j)} + \alpha_1^{(j)}\eta_1 + \cdots + \alpha_l^{(j)}\eta_l + \alpha_{11}^{(j)}\eta_1^2 + \alpha_{12}^{(j)}\eta_1\eta_2 + \cdots \alpha_{l,l-1}^{(j)}\eta_{l-1}\eta_{l-2} + \alpha_{ll}^{(j)}\eta_l^2 \tag{5.174}$$

A family of L different polynomials is selected. The number L and the order of each polynomial in the family are arbitrary design parameters.

Example 3: Three Link Manipulator.

Let each polynomial $\beta_j(\boldsymbol{\eta})$ be of the form

$$\beta_j(\eta) = \eta_1^r \eta_2^s \eta_3^t \tag{5.175}$$

where r, s, and t are nonnegative integer exponents that satisfy the inequality

$$0 \leq r + s + t \leq 3 \tag{5.176}$$

Taking all possible combinations results in having 20 different polynomials

$$\beta_0 = 1, \quad \beta_1 = \eta_1, \quad \beta_2 = \eta_2, \quad \beta_3 = \eta_3, \quad \beta_4 = \eta_1^2,$$
$$\beta_5 = \eta_1\eta_2, \ldots, \quad \beta_{18} = \eta_2\eta_3^2, \quad \beta_{19} = \eta_3^3$$

The polynomials are ordered according to increasing $r + s + t$, increasing $s + t$ (in case of equal values of $r + s + t$) and increasing t (in case of equal values of $s + t$).

Hence, a natural choice of L in this case (i.e., the case of three variate polynomials of order 3) is 20.

Let $\beta_j(\boldsymbol{\eta}_i)$ be the value of polynomial j at the robot configuration i. The following model is now fitted to the measurement data:

$$\begin{aligned} c_{k0}\beta_0(\boldsymbol{\eta}_1) + c_{k1}\beta_1(\boldsymbol{\eta}_1) + \cdots + c_{k,L-1}\beta_{L-1}(\boldsymbol{\eta}_1) &= (\Delta\eta_k)_1 \\ c_{k0}\beta_0(\boldsymbol{\eta}_2) + c_{k1}\beta_1(\boldsymbol{\eta}_2) + \cdots + c_{k,L-1}\beta_{L-1}(\boldsymbol{\eta}_2) &= (\Delta\eta_k)_2 \\ &\vdots \\ c_{k0}\beta_0(\boldsymbol{\eta}_m) + c_{k1}\beta_1(\boldsymbol{\eta}_m) + \cdots + c_{k,L-1}\beta_{L-1}(\boldsymbol{\eta}_m) &= (\Delta\eta_k)_m \end{aligned} \tag{5.177}$$

for $k = 1, \ldots, K$. c_{kj} is an unknown coefficient of polynomial j for the kth joint correction. There is a total of $K \cdot L$ unknown coefficients, or, more descriptively, there are K separate linear least-squares problems involving each L unknown coefficients and m data points. Obviously an over determining condition $m > L$ must be satisfied.

Denote by the matrix $\mathbf{B}$ the values of the polynomial at the measurement points.

$$\mathbf{B} = \begin{bmatrix} \beta_0(\boldsymbol{\eta}_1) & \cdots & \beta_{L-1}(\boldsymbol{\eta}_1) \\ \vdots & & \vdots \\ \beta_0(\boldsymbol{\eta}_m) & \cdots & \beta_{L-1}(\boldsymbol{\eta}_m) \end{bmatrix} \tag{5.178}$$

Also denote

$$\mathbf{c}_k = [c_{k0}, c_{k1}, \ldots, c_{L-1}]^T \tag{5.179}$$

$$\Delta\boldsymbol{\eta}_k = [(\Delta\eta_k)_1, \ldots, (\Delta\eta_k)_m]^T \tag{5.180}$$

Then the least-squares solutions, $\mathbf{c}_k^*$, for the unknown vectors $\mathbf{c}_k$, $k = 1, \ldots, K$ are

$$\mathbf{c}_k^* = [\mathbf{B}^T\mathbf{B}]^{-1}\mathbf{B}^T\Delta\boldsymbol{\eta}_k, \qquad k = 1, \ldots, K \tag{5.181}$$

For arbitrary selection of polynomials and measurement points there is no way to ensure that the matrix $\mathbf{B}^T\mathbf{B}$ will not become ill conditioned. This calibration method should then involve in addition to a careful selection of the family of approximating polynomials an even more careful selection of the robot measurement configurations. Both selections are mutually dependent.

The idea is to construct the polynomials to be orthogonal to each other at the measurement points. Orthogonality of two polynomials $\beta_{j1}(\boldsymbol{\eta})$ and $\beta_{j2}(\boldsymbol{\eta})$ at m_1 measurement points means that

$$\sum_{i=1}^{m_1} \beta_{j1}(\boldsymbol{\eta}_i)\beta_{j2}(\boldsymbol{\eta}_i) = 0 \tag{5.182}$$

where $m_1 \leq m$ (m is the total number of measurement points).

The orthogonalization algorithm is carried in the following manner:

Given:

1. A set of ordered measurement data points $(\boldsymbol{\eta}_i, \Delta\boldsymbol{\eta}_i)$, $i = 1, \ldots, m$
2. A family of multivariate ordered polynomials as in Example 3.

Then

Step 1: Pick up $\beta_0 = 1$.

Step 2: Let $\beta_1(\boldsymbol{\eta})$ be a linear combination of β_0 and a new term that is introduced by multiplying β_0 with one of the independent variables, say η_1.

$$\beta_1(\boldsymbol{\eta}_i) = \alpha_0\beta_0 + \beta_0\eta_1 \tag{5.183}$$

Step 3: The undetermined coefficient α_0 in Equation 5.183 is chosen such that $\beta_0(\boldsymbol{\eta})$ and $\beta_1(\boldsymbol{\eta})$ are orthogonal to each other at $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$, as follows:

$$\beta_0(\boldsymbol{\eta}_1)\beta_1(\boldsymbol{\eta}_1) + \beta_0(\boldsymbol{\eta}_2)\beta_1(\boldsymbol{\eta}_2) = 0 \tag{5.184}$$

Substitution of $\beta_0(\boldsymbol{\eta}_1) = \beta_0(\boldsymbol{\eta}_2) = 1$, $\beta_1(\boldsymbol{\eta}_1) = \alpha_0 + \eta_1^{(1)}$, $\beta_2(\boldsymbol{\eta}_2) = \alpha_0 + \eta_2^{(2)}$ into Equation 5.183 provides the unique solution for α_0:

$$\alpha_0 = \frac{\eta_1^{(1)} + \eta_1^{(2)}}{2} \tag{5.185}$$

Step 4: Let $\beta_2(\boldsymbol{\eta})$ be a linear combination of all previous polynomials, that is, $\beta_0(\boldsymbol{\eta})$ and $\beta_1(\boldsymbol{\eta})$ and a new term that is introduced by multiplying a previous polynomial by one of the independent variables [maintaining a consistent ordering of $\beta_i(\boldsymbol{\eta})$ as in Example 3].

Therefore, choosing a previous polynomial to still be β_0, however, this time multiplied by η_2, we get

$$\beta_2(\boldsymbol{\eta}) = \alpha_1\beta_0 - \alpha_2\frac{\eta_1^{(1)} + \eta_1^{(2)}}{2}\eta_1 + \beta_0\eta_2 \tag{5.186}$$

Step 5: The parameters α_1, α_2 are adjusted to ensure pairwise orthogonality with respect to previous polynomials. That is, orthogonality of β_0 and β_2 at the points $\boldsymbol{\eta}_1$, $\boldsymbol{\eta}_2$, and $\boldsymbol{\eta}_3$:

$$\sum_{i=1}^{3} \beta_0(\boldsymbol{\eta}_i)\beta_2(\boldsymbol{\eta}_i) = 0 \tag{5.187}$$

together with orthogonality of β_1 and β_2 at the same three measurement configurations.

$$\sum_{i=1}^{3} \beta_1(\eta_i)\beta_2(\eta_i) = 0 \tag{5.188}$$

A unique solution (α_1, α_2) results.

Next Steps: The algorithm continues in the same manner. For each new measurement point a new polynomial is created, such that it is orthogonal to every one of the previous polynomials

As a result of the orthogonalization algorithm, the matrix $\mathbf{B}^T\mathbf{B}$ becomes diagonal.

A final improvement to the orthogonalizing algorithm is a normalizing of each polynomial as it is obtained. This is accomplished by multiplying the polynomial by a constant γ_j so that

$$\gamma_j \sum_{i=1}^{m} \beta_j^2(\eta_i) = 1 \tag{5.189}$$

The set of polynomials, in such a case, is said to be orthonormal and the $\mathbf{B}^T\mathbf{B}$ matrix becomes the identity matrix. An explicit least-squares solution (Equation 5.181) immediately results.

So far we assumed an arbitrary set of measurement points. The least-squares solution is highly sensitive to the selection of data points. In other words, using the resulting approximating polynomials (Equation 5.174) for inverse calibration may exhibit different levels of error depending on the set of measurement points that is used to construct the approximating polynomials. The use of a uniformly spaced grid of measurement points, in the robot joint space, may turn out to be highly suboptimal.

The problem of optimal spacing of measurement points is a classical numerical analysis topic referred to in the literature under the title "Chebyshev polynomials" (see for example Isaacson and Keller [9] or Fox and Parker [4]). Some of the key ideas of the one-dimensional case of approximating a function $f(x)$ by a polynomial of degree n are outlined next. Extensions to multiple dimension, that is, the fitting of multivariate polynomials to functions of several variables, are straightforward. In other words, once it is understood how to space the data points along one axis (in joint space), the same spacing strategy can be employed independently along each of the other axes.

Let $f(x)$ be an arbitrary continuous function of a one-dimensional variable x. Suppose that we decide to approximate $f(x)$ by a polynomial of degree n, $p_n(x)$, with a leading coefficient 1. The degree of the polynomial n is a fixed preselected parameter. The problem is to find the other n coefficients of $p_n(x)$ to minimize the error defined as

$$d(f, p_n) = \|f(x) - p_n(x)\|_\infty = \max\{|f(x) - p_n(x)|, \quad \text{for } a \le x \le b\} \tag{5.190}$$

where $[a, b]$ is the interval of interest of x. A polynomial $p_n^*(x)$ that minimizes $d(f, p_n)$ is the "best approximation" among all polynomials of degree of at most n.

It was observed by the Russian mathematician Chebyshev that such a best approximating polynomial is unique and the related error function $f(x) - p_n^*(x)$ must attain the extreme values $\pm d(f, p_n^*)$, with alternate changes of sign, at least $n + 2$ times in $[a, b]$, including the end points of the interval. To get a better understanding of why this is so, consider the important special case of $f(x) = 0$. Suppose that $q_n^*(x)$ is another polynomial of degree n with leading coefficient 1 that has a smaller extreme value, that is $d(f, q_n^*) \leq d(f, p_n^*)$. The difference $p_n^*(x) - q_n^*(x)$ is another polynomial of degree $n - 1$ that has alternating positive and negative values at $n + 1$ points at least, or in other words, it has n zeros, which is impossible. Therefore $q_n^*(x) = p_n^*(x)$.

The most obvious functions with successively equal and opposite values are the trigonometric functions $\sin\theta$ and $\cos\theta$, with equal and opposite values of ± 1 at $n + 1$ points in $0 \leq \theta \leq \pi$, including the end points. It is deduced that the required unique best approximating polynomial in the special interval of $[a, b] = [-1, 1]$, creates an error function that is a multiple of $t_n(x)$, where

$$t_n(x) = \cos(n \cos^{-1} x) \tag{5.191}$$

$t_n(x)$ is known as the Chebyshev polynomial. We notice that the coefficient of x^n in $t_n(x)$ is 2^{n-1}. Therefore the best approximating polynomial $p_n^*(n)$ with a leading coefficient of 1 should have a related error function:

$$f(x) - p_m^*(x) = 2^{1-n} t_n(x) \tag{5.192}$$

A simple change of variables

$$y = \tfrac{1}{2}[(b - a)x + (a + b)] \tag{5.193}$$

converts the approximation of $f(x)$ over $-1 \leq x \leq 1$ into a related approximation of $g(y)$ over $a \leq y \leq b$, for arbitary intervals $[a, b]$.

An obvious consequence of Equation 5.192 is that $p_n^*(x)$, the best approximating polynomial of degree n, is actually *equal* to $f(x)$, the function it approximates, at $n + 1$ distinct point $x_0, \ldots, x_n$. These points correspond to the zeros of the Chebyshev polynomial $t_n(x)$.

For arbitrary continuous function $f(x)$, the above is still not enough to actually construct $p_n^*(x)$. Some more insight can be obtained if $f(x)$ happens to have $n + 1$ continuous derivatives over the interval of interest $[a, b]$. In such a case there exists $n + 1$ distinct points, $x_0, x_1, \ldots, x_n$ in $[a, b]$ such that

$$f(x) - p_n^*(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_n)}{(n + 1)!} f^{(n+1)}(\zeta) \tag{5.194}$$

where $f^{(n+1)}(\cdot)$ denotes the $(n + 1)$th derivative of $f(x)$, and $\zeta(x)$ is a midpoint in the interval $[\min(x, x_0, \ldots, x_n), \max(x, x_0, \ldots, x_n)]$.

The case $f^{(n+1)} = \text{constant}$ corresponds to the case where $f(x)$ is a polynomial of degree of at most $n + 1$. In this special case the error expressed in terms of

Equation 5.194 can be minimized by choosing the points $x_0, x_1, \ldots, x_n$ such that the polynomial $(x - x_0)(x - x_1)\cdots(x - x_n)$ has the smallest possible maximum absolute value in the interval $[a, b]$. It becomes obvious now why the particular case of $f(x) = 0$ was said to be so important. In the robot calibration problem, the functions $f(\eta_1, \ldots, \eta_K)$ are not known explicitly. All is known is a set of interpolation points ($\boldsymbol{\eta}_i$ and the measured joint correction in each such configuration).

The optimal spacing problem formulation can now be done as follows:

Given an interval $[a, b]$ for x, find a set of interpolation points $x_0, x_1, \ldots, x_n$ in the interval such that the maximum of the absolute value of $|(x - x_0)(x - x_1) \cdots (x - x_n)|$ in the interval is minimized.

The optimal solution $x_0^*, x_1^*, \ldots, x_n^*$ is then used to define the measurement points at which measurement readings $f(x_0^*), f(x_i^*), \ldots, f(x_n^*)$ are taken and a least-squares fit of a polynomial of degree $n + 1$ is performed.

The solution to the optimal spacing problem is readily given in terms of the zeros of the Chebyshev polynomial. Thus:

$$x_k^* = \frac{1}{2}\left[(b - \alpha)\cos\left(\frac{2k + 1}{n + 1}\cdot\frac{\pi}{2}\right) + (a + b)\right]; \qquad k = 0, 1, \ldots, n \tag{5.195}$$

One can then show that the minimal value of the largest deviation of $|(x - x_0)\ldots(x - x_n)|$ from zero is $2^{-n}|0.5(b - a)|^{n+1}$.

The effect of Chebyshev spacing is to place more points near the boundaries of the interval of interest and fewer in the interior. Application of all these ideas to robot calibration is not straightforward. By applying Chebyshev spacing in the joint space as one should, the robot user is not left with a clear knowledge of which part of the robot workspace has been calibrated. By applying Chebyshev spacing in world coordinates, the optimality of the spacing algorithm, in the sense explained above, is no longer assured. As an "ad hoc" solution, reported by Shamma [12], Chebyshev spacing was performed in the world coordinates that best suit the geometry of the individual manipulator to avoid unreachable "pockets." For a cartesian manipulator this would have involved spacing along the XYZ axes. For the PUMA 560 simulations [12], spacing was performed in cylindrical coordinates, as shown in Figure 5.4.

Example 4: Calibration of the First Three Degrees of Freedom of the PUMA 560 Robot (see Whitney and Shamma [11, 12]).

The calibration procedure for the waist, shoulder, and elbow of the PUMA 560 robot can be summarized as follows:

Step 1: Define a region in the robot's workspace and generate a set of training points via Chebyshev spacing (see Figure 5.4.)

Step 2: Construct a set of orthonormal three variate polynomials in the variables η_1, η_2, and η_3.

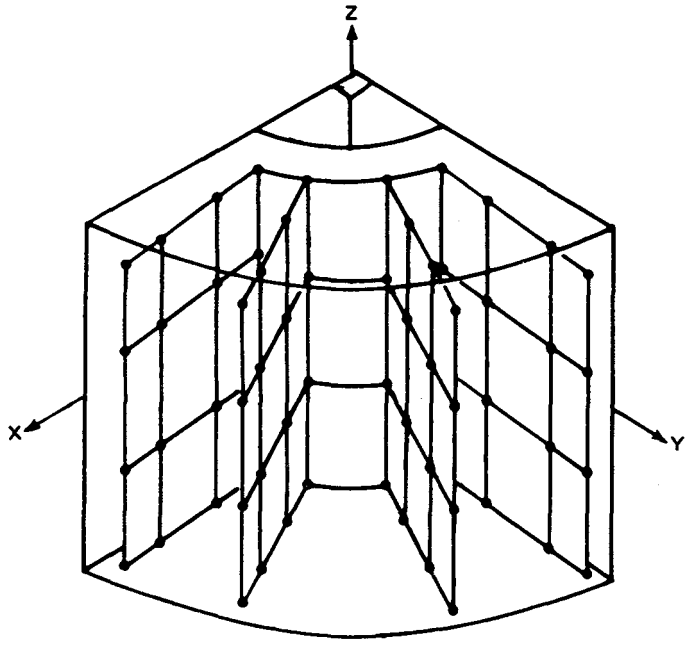


Figure 5.4. Data points generated by Chebyshev spacing. Reprinted with permission of MIT and the author [12].

Step 3: At the above training points find the required joint corrections. To find the encoder corrections necessary to drive the manipulator to the desired world coordinate position involves the following experimental procedure:

1. Let the desired test point position be $\mathbf{x}_i^{(d)}$, which corresponds to a joint position vector $\boldsymbol{\eta}_i^{(n)}$, according to the robot nominal inverse kinematics. Apply the command $\boldsymbol{\eta}_i^{(n)}$, and measure the actual end point location, $\mathbf{x}_i^{(a)}$, using the measurement device.
2. Manually perturb the joints until the manipulator end point is in the desired training point $\mathbf{x}_i^{(d)}$. Record the joint readings $\boldsymbol{\eta}_i^{(a)}$. Then $\Delta\boldsymbol{\eta}_i = \boldsymbol{\eta}_i^{(a)} - \boldsymbol{\eta}_i^{(n)}$.

Step 4: Solve for the coefficients that give the polynomials the best fit to the data. There will be three such sets of coefficients, one set for each joint.

The procedure was applied to a simulated PUMA. Sixty-four data points were generated. Hence, the dimension of the **B** matrix was 64 × 20. Chebyshev spacing was performed in $R\Phi Z$ cylindrical coordinates with four points along each axis. The ranges were as follows

$$300 \leq R \leq 860 \text{ mm}$$

$$0 \leq \Phi \leq 90^\circ$$

$$-500 \leq Z \leq 500 \text{ mm}$$

which covers about a quarter of the manipulator's workspace. In the simulated nominal and actual models, accuracy errors were found in all the data points. The average positioning error was 1.6 mm with standard deviation of 0.51 mm and maximum difference of 2.5 mm. After calibration, the positioning error between the actual robot model and the calibrated robot model reduced to an average of 0.12 mm at the data points, with standard deviation of 0.06 mm and maximum difference of 0.3 mm. In addition the calibration was also tested at 50 randomly selected points in the calibrated region with roughly similar accuracy results.

Direct extension of the method presented for calibrating a 3 DOF manipulator ($N = 3$) to the case of a general manipulator may be very cumbersome computationally. For instance, the use of third-order, six-variate polynomials requires 84 terms (compared to 20 terms as shown in the example), thus requiring a very large and probably impractical number of data points.

One may start by calibrating the first three degrees of freedom of the manipulator, keeping the wrist joints locked in their zero positions. One cannot, however, calibrate the wrist in the same manner, keeping the first three joints locked, since the experimental procedure for finding the joint corrections may not apply in this case. The desired wrist training positions may not be reachable when the first three joints are locked and only the wrist joints are varied.

A procedure for wrist calibration, based on measurements of end point position only is proposed in Shamma's work [11, 12]. It is assumed that the robot's first three joints have already been calibrated. Thus, when a robot end effector is commanded to go to a certain desired position and orientation represented by the transformation $\mathbf{X}_d$ and ends up at an actual location $\mathbf{X}_a$, the actual position error, $\mathbf{e}_p$, defined as

$$\mathbf{X}_a - \mathbf{X}_d = \begin{bmatrix} \Delta\mathbf{R}_{3\times 3} & \mathbf{e}_p \\ \mathbf{0} & 0 \end{bmatrix} \tag{5.196}$$

must depend only on the wrist joint values, if $\mathbf{e}_p$ is measured with respect to a cordinate frame at the desired end point location:

$${}^{(T)}\mathbf{e}_p = \mathbf{e}_p(\eta_4, \eta_5, \eta_6) \tag{5.197}$$

The left superscript (T) indicated that $\mathbf{e}_p$ is measured with respect to the tool coordinate frame. The relationship between $\mathbf{X}_a$ and $\mathbf{X}_d$ is through a translation transformation denoted as the "tool compensation transformation" ${}^{(T)}\mathbf{T}_c$

$$\mathbf{X}_a = \mathbf{X}_d {}^{(T)}\mathbf{T}_c \tag{5.198}$$

Thus, the XYZ compensation translations are found from the fourth column of $\mathbf{X}_d^{-1}\mathbf{X}_a$. For this, it is enough to measure only the positon of $\mathbf{X}_a$ and orientation may be ignored.

For the same values of the first three joints, the tool compensation transformation is found experimentally for different sets of wrist angles. One can now construct three variate polynomials to approximate the compensation XYZ translations to the wrist joint positions at a given end effector position.

Such polynomials need to be constructed at different values of the robot first three joints.

5.4.2 Cerebellar Model Articulation Controller (CMAC)

As discussed in the previous sections, there are a number of different approaches to solving the inverse kinematics problem for a robot manipulator. For a solution to be useful for real time trajectory computations, it must be both accurate and computationally efficient. Many of the algorithms described earlier have placed a primary emphasis on accuracy and are practical only for off-line trajectory generation. One approach to compromising between speed and accuracy would be to divide the workspace of the robot into a number of discrete areas and then use a numerical procedure to determine the inverse kinematic solution for each area. These solutions would then be stored in nonvolatile memory and could be quickly recalled by the robot controller. Unfortunately, this simple approach requires a prohibitively large amount of memory. For example, consider a typical six axis robot with a workspace of 30 × 30 × 30 in. If we assign a resolution of 0.1 in. to each Cartesian position coordinate and 1.2° on each orientation coordinate, the memory required would be roughly 0.729×10^{15} words. Clearly this is far in excess of what is available with today's technology. These large memory requirements can possibly be reduced, however, by applying a technique based on the concepts of a neural network. The cerebellar model articulation controller (CMAC) was originally developed by James Albus [1, 2] to model the function of the cerebellar cortex of the brain, but its efficient use of memory and property of generalization make it useful as a general purpose function approximator.

5.4.2.1 Description of CMAC To clarify the description of CMAC, a simple example will be developed. A pictorial representation of this two input example is shown in Figure 5.5. The input space consists of a two-dimensional surface, which may be represented with a Cartesian coordinate system XY. Any input may be represented as a vector, $\mathbf{s}$, such that

$$\mathbf{s} = [X, Y]^T \tag{5.199}$$

It is important to note that the input space consists of discrete elements, meaning that the input axes have a finite resolution. The accuracy of CMAC as a function approximator is compromised by this discretization. On the other hand, finite

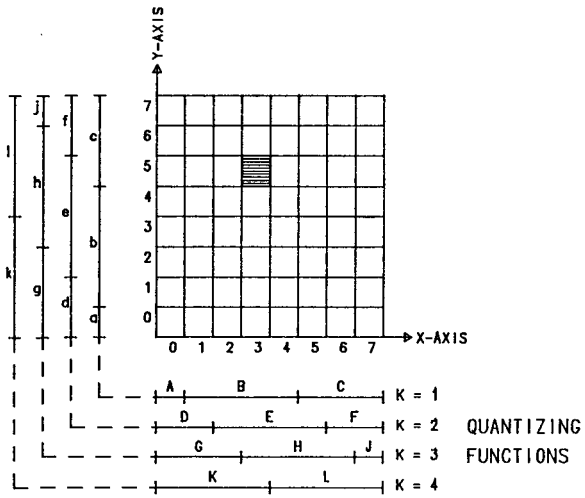


Figure 5.5. An example of a CMAC workspace.

resolution is a reality for most robots due to hardware limitations and the CMAC can be tailored to match this.

Basically, CMAC is a mapping that encodes an input vector into a set of pointers that selects a set of weights from subtables. The selected weights are then summed to generate the output vector. The encoding of the input vector is accomplished with "quantizing functions" applied to each of the input axes. In our example, four quantizing functions are used. Each quantizing function is represented by the lines labeled $k = 1$ through $k = 4$ in Figure 5.5. Each quantizing function is the encoder that points to one subtable. The subtables for our example are illustrated in Figure 5.6. For example, given the input vector $\mathbf{s} = [3, 5]^T$, the weights selected from the subtables are the set (Bc, Ee, Hh, Kl). Each element of this set has a weight associated with it and the output is calculated as the sum of each of these weights. This process for the input vector $\mathbf{s} = [3, 5]^T$ is illustrated in Table 5.8.

CMAC selects a unique set of weights for each input vector. Another input vector that lies in the neighborhood of the first would map to a different set, but would contain many weights selected by the first vector. This may be rephrased by saying that points that are close together in the input space have output values that are similar. This is the basis of the property known as "generalization."

To set up a CMAC model for a particular problem, several elements of the process must be specified. First, an appropriate resolution must be chosen for the input space. This choice is based on the requirements of the particular application. It must be remembered, however, that the finer the resolution of the input space, the greater the memory requirement for the CMAC model. The second choice is the number of quantizing functions that will be used. The maximum allowable number of quantizing functions is equal to the number of resolution blocks along one axis of the input space (R) minus one. The minimum

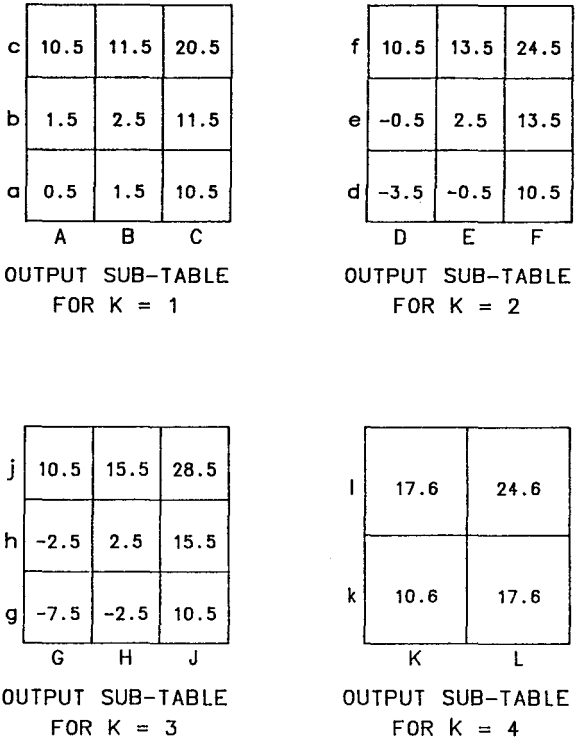


Figure 5.6. Output subtables for the CMAC example problem.

number of quantizing functions is one. One quantizing function represents a "one-to-one" mapping and uses the maximum amount of memory. The more quantizing functions used, the less memory will be required. It seems reasonable to assume that as the amount of required memory is decreased, the accuracy of the CMAC output will degrade. It is, therefore, important to establish a relationship between the number of quantizing functions (memory usage) and CMAC accuracy. The final step in establishing a CMAC model is to determine the weights that go in the subtables. Albus [1, 2] proposed a "teaching" scheme whereby the weights are determined iteratively by trial and error. This technique is reviewed in the following paragraphs.

TABLE 5.8. Subtable Weights

Subtable	Location	Weight Value
1	Bc	11.5
2	Ee	2.5
3	Hh	2.5
4	Kl	17.6
	Output	34.1

5.4.2.2 CMAC and Learning So far, no mention has been made of where the weights in the subtables come from. This is the crux of memory-based systems such as CMAC. For the cerebellar cortex, the "weights" are thought to evolve from past experience. In this light, Albus [1,2] proposed an iterative teaching scheme for CMAC that could loosely be described as learning from experience by reinforcement and punishment. This technique, which is analogous to a simple control system, uses proportional feedback to correct an error in the output of the system. When CMAC generates an output, it is compared to a "desired" or correct output, which may be thought of as a reference. The difference between these two values is the error, and this is used to correct the values in the subtables. The weights are adjusted according to the size of this error signal. As proposed by Albus, the error is equally divided among and algebraically added to the weights that were used to generate the output.

A totally empty or blank CMAC can learn very quickly because of generalization. An input vector will access a number of weights to generate an output, and the same set of weights will be corrected by the error feedback mechanism. Theoretically, therefore, only a few experiences are needed to teach CMAC to a level where, in general, it will respond with a reasonable output across the entire input space. This phenomenon will be illustrated in a later example.

Two experiences close together in the input space would teach a large number of the same weights. In each case, the weights would be adjusted to reach zero error for the input in question, while possibly adversely affecting the result for the other input. Thus, the learning could very well be oscillatory. This tendency toward oscillatory learning may be reduced by multiplying the correction during each learning experience by some gain, g. The gain will modify the effect that each learning experience has on the particular set of groups that is being taught. The correction applied to each weight involved in a learning experience would be given by

$$\Delta = g\left[\frac{(\text{Desired output}) - (\text{CMAC output})}{Q}\right] \tag{5.200}$$

where Δ is the weight correction, g is the teaching gain, and Q is the number of quantizing levels.

Learning can be a continuous process as it is not expensive in terms of time if a "correct" or reference output is readily available. In this way, systems that change slowly, due to wear, for example, could automatically remain accurate without the need for recalibration.

5.4.2.3 An Example Using CMAC To investigate the application of CMAC to the solution of the inverse kinematic equations for a robot manipulator, a simple example will be investigated. A two link arm with revolute joints having parallel axes and a planar workspace was used. This geometry is similar to that of a SCARA robot and is illustrated in Figure 5.7. To avoid multiple solutions, the joint angles are constrained to

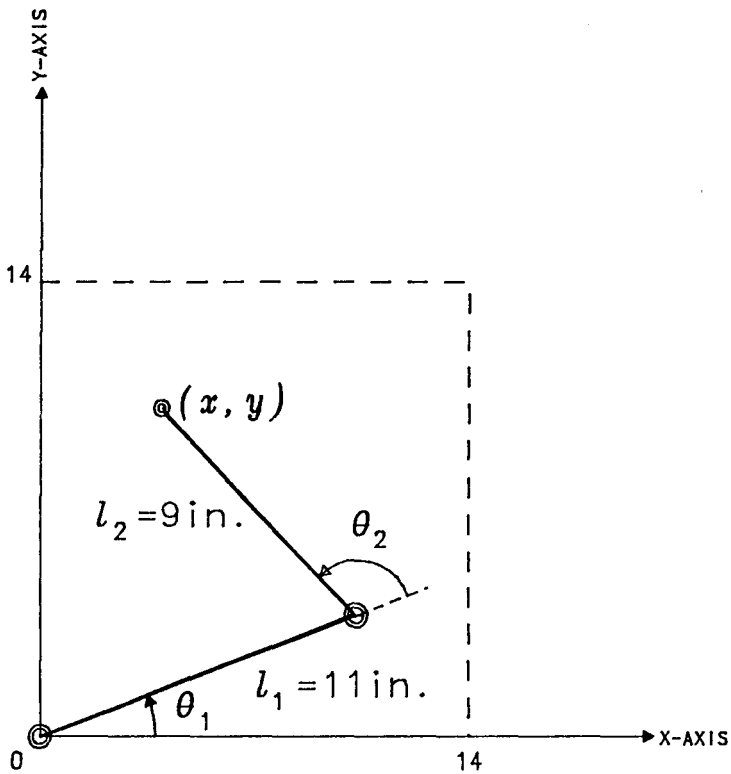


Figure 5.7. Two link robot in the CMAC example.

$$-90° \leq \theta_1 \leq 90° \tag{5.201}$$

$$0° \leq \theta_2 \leq 180° \tag{5.202}$$

and the workspace is limited to

$$2'' \leq x < 14'' \tag{5.203}$$

$$2'' \leq y < 14'' \tag{5.204}$$

This definition simplifies the application of CMAC by keeping the input space square and excluding the region defined by the loci of points $(x^2 + y^2)^{1/2}$, at which the functions for θ_1 and θ_2 become discontinuous.

The inverse kinematic solution for this mechanism is easily determined and may be written

$$\theta_1 = \tan^{-1}(y/x) - \cos^{-1}[(r^2 + l_1^2 - l_2^2)/(2rl_1)] \tag{5.205}$$

$$\theta_2 = \cos^1[(r^2 - l_1^2 - l_2^2)/l_1 l_2)] \tag{5.206}$$

where $r = (x^2 + y^2)^{1/2}$.

From the point of view of CMAC, the problem is a two-input, two-output mapping where inputs x and y range from 2 to 12.9. The input axes were divided into 120 resolution units 0.1 in. wide, so that the CMAC input variables could be expressed as integers from 0 to 119. These discrete CMAC input variables, iX and iY, may be related to the cartesian variables by the equations

$$iX = \text{nint}[10(x - 2)] \tag{5.207}$$

$$iY = \text{nint}[10(y - 2)] \tag{5.208}$$

where "nint" rounds a rational number to the nearest integer.

With this scheme for specifying the input vector, the mapping from input space to output tables may be expressed as follows:

$$xM = \text{int}[(iX + Q - K)/Q] + 1 \tag{5.209}$$

$$yM = \text{int}[(iY + Q - K)/Q] + 1 \tag{5.210}$$

where "int" represents integer truncation and where xM and yM give the location of the weight to be used in the output subtable K. The variable xM represents the row number starting from the bottom row and yM is the column number starting from the left column.

The choice for Q, the number of quantizing functions, is arbitrary and was set at 10. This results in output subtables of size 12 × 12 and a memory requirement of 1440 words for each output variable. A one to one mapping for the same problem would require 14,400 words. Thus, the savings in memory is significant, with CMAC requiring only 10% of the memory required for a direct table lookup.

The outputs of the CMAC were calculated as follows:

$$\theta_i = \sum_{K=1}^{Q} W_K(xM, yM), \qquad i = 1, 2 \tag{5.211}$$

where $W_K(xM, yM)$ is the weight in subtable K pointed to by xM and yM. It should be noted that there will be a different set of weights for each output, θ_i. Since the closed form solution for the inverse kinematic equations exists, this was used to provide the reference output for teaching the CMAC. Only one output, θ_1, will be examined although both were taught. The reasons for this are that θ_1 is the more complicated function of x and y, and, as it turned out, θ_1 has worse errors than θ_2 when computed by CMAC. Lastly, as a function of two variables can be viewed as a plot, graphs are used to study the trends in teaching.

A few preliminary tests were made to examine the response of CMAC to teaching. Starting with a set of blank weight tables, the system was taught the functions for θ_1 and θ_2 at a few points distributed evenly throughout the input space. Figures 5.8 through 5.11 show how rapidly θ_1 approached the shape of the desired function, although the absolute errors are quite high. This set of results is representative of many others that indicate that CMAC is potentially

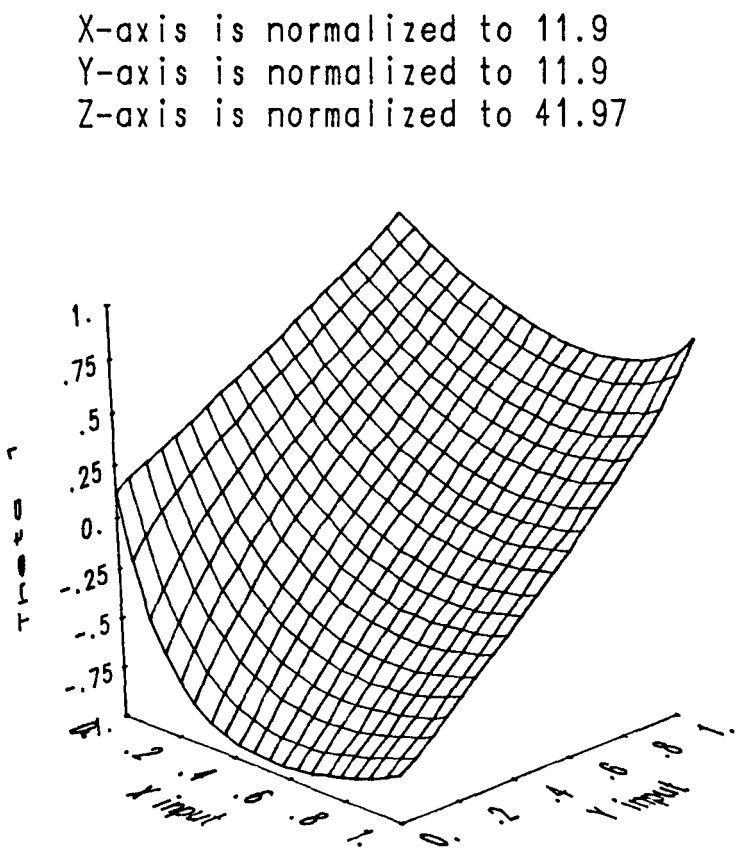


Figure 5.8. Plot of analytical surface for θ_1 as a function of X and Y.

a viable technique for solution of the inverse kinematic equations, if the error can be brought to within an acceptable bound.

In an effort to reduce the error, a random number generator was used to compute a set of input vectors to be used both as teach and test points for CMAC. In all the cases run, the teaching was considered successful when the CMAC was able to compute 100 consecutive answers for θ_1 and θ_2 that were both within a preset error tolerance. The seed for the random number generator was kept the same so as to provide identical learning histories, with a maximum of 500,000 points being taught for all cases.

Two series of tests were conducted. In the first, the teach gain, g, was varied from 0.1 to 2.0 while convergence to an error tolerance of 1.0° was observed. Figure 5.12 shows that the fastest convergence occurs for teach gains in the range of 1.0 to 1.3. For $g < 1$, the system reacts sluggishly and appears to learn very slowly. For $g > 1.3$, the learning appears to be oscillatory, which also slows

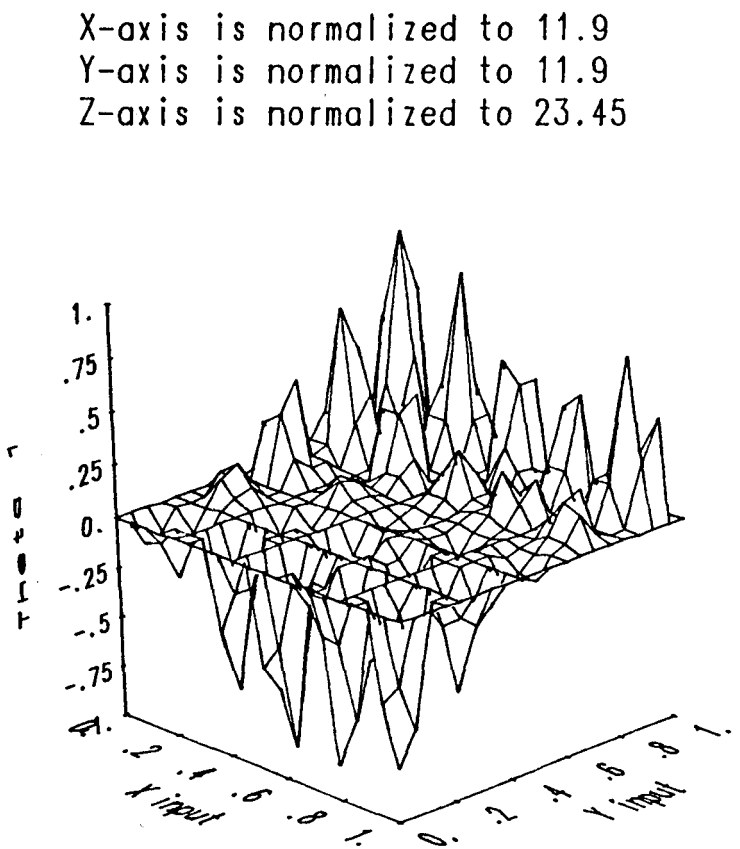


Figure 5.9. θ_1 surface generated by CMAC after being taught at 25 locations.

down the learning experience. Convergence did not occur within 500,000 experiences for $g > 2.0$.

The second series of tests were run with the above results in mind. The teach gain was set at 1.2 and the number of experiences required to reach a certain error level was recorded. Figure 5.13 is a plot of these data and illustrates that as the error tolerance decreases, the number of experiences required to meet the tolerance increases asymptotically. Thus, we would expect that with infinite teaching, the CMAC will converge to some minimum error, which will not be zero.

It is important to note that errors in the output will arise from two basic sources. First there is an error due to the quantization of the input space. This component of the error will be referred to as discretization error. There is also some error that is inherent in the CMAC process of locating and summing weights, which we will refer to as CMAC error. In other words, if the actual relationship between the input and output was discrete and there were no discretization error, the CMAC model would still not yield a perfect relationship.

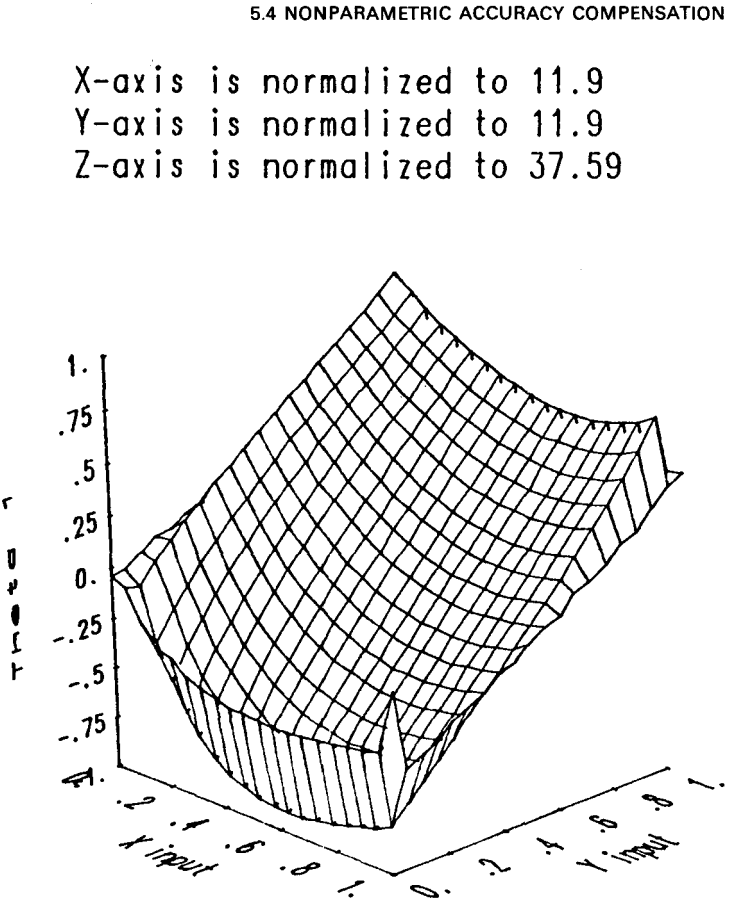


Figure 5.10. θ_1 surface generated by CMAC after being taught at 114 locations.

In an effort to examine the relative magnitude of the quantization error and the CMAC error, the following test was run. The maximum error for θ_1, given a CMAC error in θ_2, that could be obtained while still keeping the linkage tip inside a given input element was determined at a number of points distributed over the input space. In other words, the maximum possible discretization error in θ_1 was determined given the CMAC error in θ_2 for that input. Figure 5.14 is an illustration of this maximum discretization error throughout the input space. This plot shows only the maximum positive error. A similar plot exists for the maximum negative error. It can be seen in Figure 5.14 that, for the most part, the discretization error seems to be the same order of magnitude as the errors inherent to CMAC. In this light, CMAC seems to be successful in this application.

CMAC seems to be a viable approximation method with reasonable accuracy for the problem examined. The accuracy can be improved by increasing the resolution of the workspace, which would require more memory. The above

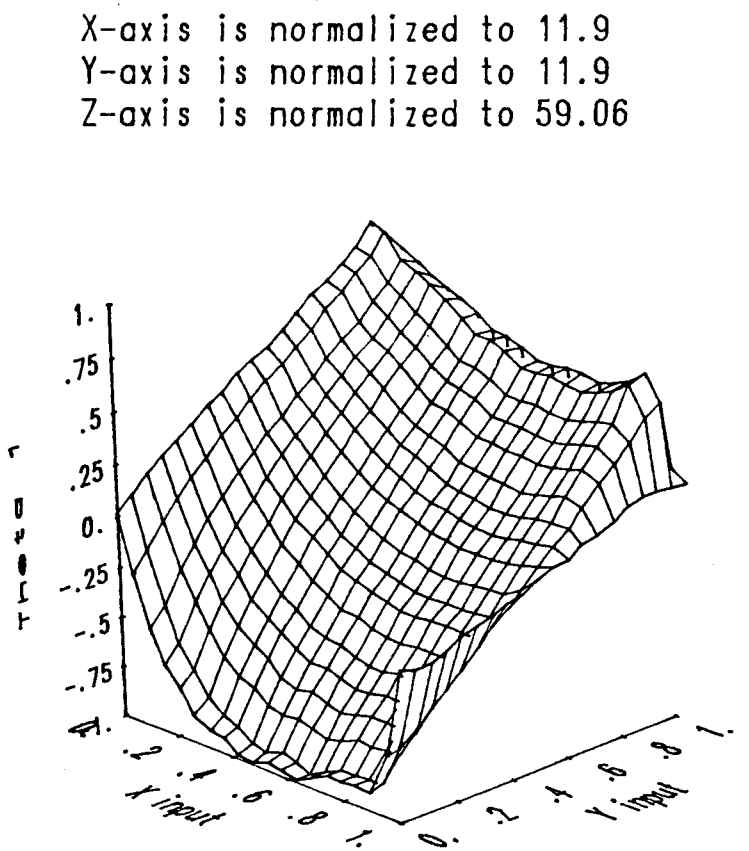


Figure 5.11. θ_1 surface generated by CMAC after being taught at 625 locations.

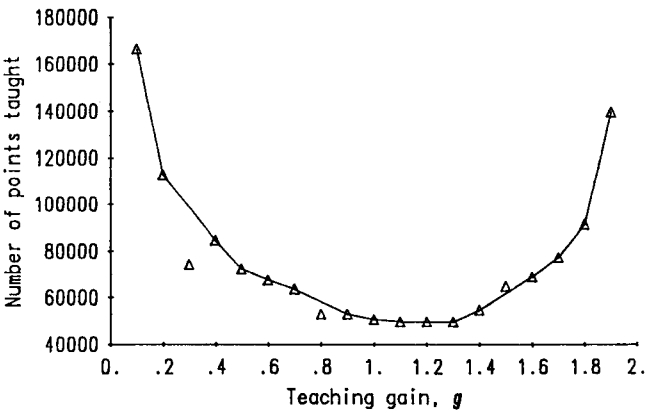


Figure 5.12. Teaching effectiveness as a function of teach gain, g.

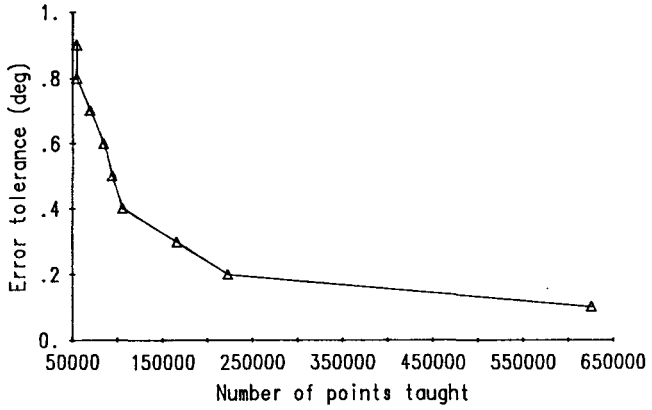


Figure 5.13. Accuracy as a function of number of points taught.

example indicates that an approach such as CMAC can be useful for robots having only a few degrees of freedom. Its utility for more complicated geometries or for higher resolutions is still an open question. Although topics such as automated or one-step teaching and higher order CMACs are being investigated, the ultimate utility of this approach is still very much in debate.

X-axis is normalized to 11.9
Y-axis is normalized to 11.9
Z-axis is normalized to 1.1

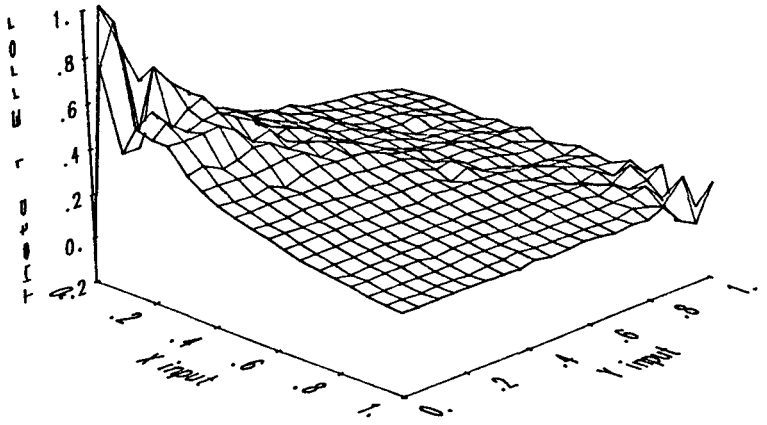


Figure 5.14. Maximum discretization error in θ_1 (degrees).

5.5 CONCLUSION

This chapter presented a comprehensive treatment of model-based kinematic accuracy compensation methods down to the smallest algorithmic details. The methods vary in their computational complexity starting from a simplistic inverse Jacobian technique to a more complex optimal control strategy.

In comparison to model-based compensation methods, which have attained a significant level of maturity, nonparametric methods, which are essential to the addressing of nongeometric accuracy errors, are still at an early stage of development. More development work is required before the applicability of these methods will become well understood. This chapter presented an introduction to the key ideas behind Shamma and Whitney's inverse calibration methods that use polynomial approximation functions. The use of CMAC was also demonstrated for the inverse kinematics of a simple robot.

The actual implementation of the accuracy compensation phase may vary from one robot to another depending on features of the machine control software. Not every commercially available robot controller or control language allows the modification of taught or preprogrammed joint commands. The same is true about the robot forward or inverse kinematic models implemented as part of the robot control software. The control software often consists only of the nominal kinematic model. Calibration awareness when originally designing the manipulator controller is important. A means should be included to allow the robot user to adjust the parameters in the kinematic model and to implement user defined accuracy compensation such as has been described in this chapter. Calibration compensation awareness should also be exercised by robot users when programming an application through implementation of joint level safety margins near joint travel boundaries and robot singularity zones.

REFERENCES

[1] J. S. Albus. Data storage in the cerebellar model articulation controller (CMAC). *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, 97:228–233, September 1975.

[2] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, 97:220–227, September 1975.

[3] J. J. Craig. *Introduction to Robotics—Mechanics and Control.* Addison Wesley, Reading, MA, 1986.

[4] L. Fox and I. B. Parker. *Chebyshev Polynomials in Numerical Analysis.* Oxford University Press, London, 1968.

[5] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Sensing, Vision, and Intelligence.* McGraw-Hill, New York, 1987.

[6] Samad A. Hayati and Gerald P. Roston. Inverse kinematic solution for near-simple robots and its application to robot calibration. In *Recent Trends in Robotics:*

Modeling, Control, and Education, pp. 41–50. Elsevier Science Publishing Co., Amsterdam, 1986.

[7] J. M. Hollerbach. A survey of kinematic calibration. In Robotics Review, pp. 208–242. MIT Press, Cambridge, MA, 1988.

[8] M. Z. Huang and A. Gautam. An algorithm for on-line compensation of geometric errors in industrial robots. In *Proceedings of the Second Conference on Recent Advances in Robotics*, pp. 174–177, Florida Atlantic University, Boca Raton, Florida, May 1989.

[9] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods.* John Wiley and Sons, New York, 1966.

[10] Richard P. Paul. *Robot Manipulators: Mathematics, Programming, and Control.* MIT Press, Cambridge, MA, 1981.

[11] J. S. Shamma and D. E. Whitney. A method for inverse robot calibration. *Journal of Dynamic Systems, Measurement, and Control*, 109(1):36–43, March 1987.

[12] Jeff S. Shamma. *A Method for Inverse Robot Calibration.* Master's thesis, Massachusetts Institute of Technology, January 1985.

[13] W. K. Veitschegger and Chi-Haur Wu. Robot calibration and compensation. *IEEE Journal of Robotics and Automation*, RA-4(6):643–656, December 1988.

[14] M. I. Vuskovic. Compensation of kinematic errors using kinematic sensitivities. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 745–750, Scottsdale, Arizona, May 1989.

[15] K. J. Waldron, S. L. Wang, and S. J. Bolin. A study of the Jacobian matrix of serial manipulators. *Transaction of the ASME Journal of Mechanisms, Transmission, and Automation in Design*, 107:238–238, June 1985.

[16] D. E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, 94(4):303–309, December 1972.

[17] H. Zhuang. *Kinematic Modeling, Identification, and Compensation of Robot Manipulators.* Ph.D. thesis, Florida Atlantic University, December 1989.

[18] H. Zhuang, F. Humano, and Z. S. Roth. Optimal design of robot accuracy compensators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 751–756, Scottsdale, Arizona, May 1989.

[19] H. Zhuang, F. Humano, and Z. S. Roth. Optimal solution to the neighboring inverse kinematics. In *Proceedings of the IEEE Conference on Decision and Control*, pp. 2284–2285, Austin, Texas, December 1988.

[20] H. Zhuang, F. Humano, and Z. S. Roth. Analysis and design of robot accuracy compensators C.Y. Ho and George Zobrist, Eds. In *Progress in Robotics and Intelligent Systems* 4: Chapter 2. Ablex Publishing Co., Norwood, New Jersey 1991 (in press).

CHAPTER 6

CASE STUDY

In this chapter, we will discuss the calibration of a PUMA 560 robot in a laboratory environment. This robot is illustrated in Figure 6.1. The purpose of this case study is to demonstrate the application of the techniques discussed in Chapters 1 through 5 to an actual manipulator. We will begin this study by developing a model for the robot. Both the modified Denavit–Hartenberg (DH) and the zero reference position approaches will be demonstrated. The measurement of pose data will then be described. The measurement system used is a small coordinate measuring machine. The acquired pose data will then be applied in a nonlinear least-squares identification procedure to determine the optimal set of model parameters for both models being used. To assist in comparison of the two models, the parameters identified for the zero reference position model will be converted to the DH convention. The chapter concludes with an assessment of the results of the calibration.

6.1 MODELING

The obvious first step in the modeling process is the selection of a valid model. As long as the chosen model meets the criteria of completeness, proportionality, and equivalence, the selection of a model is rather arbitrary. To illustrate this idea, we will present both the modified DH model and the zero reference position model for the robot under study.

6.1.1 Modified Denavit–Hartenberg Model

The model development is begun by defining an appropriate set of coordinate systems. A workspace coordinate system or base frame is defined first. This frame

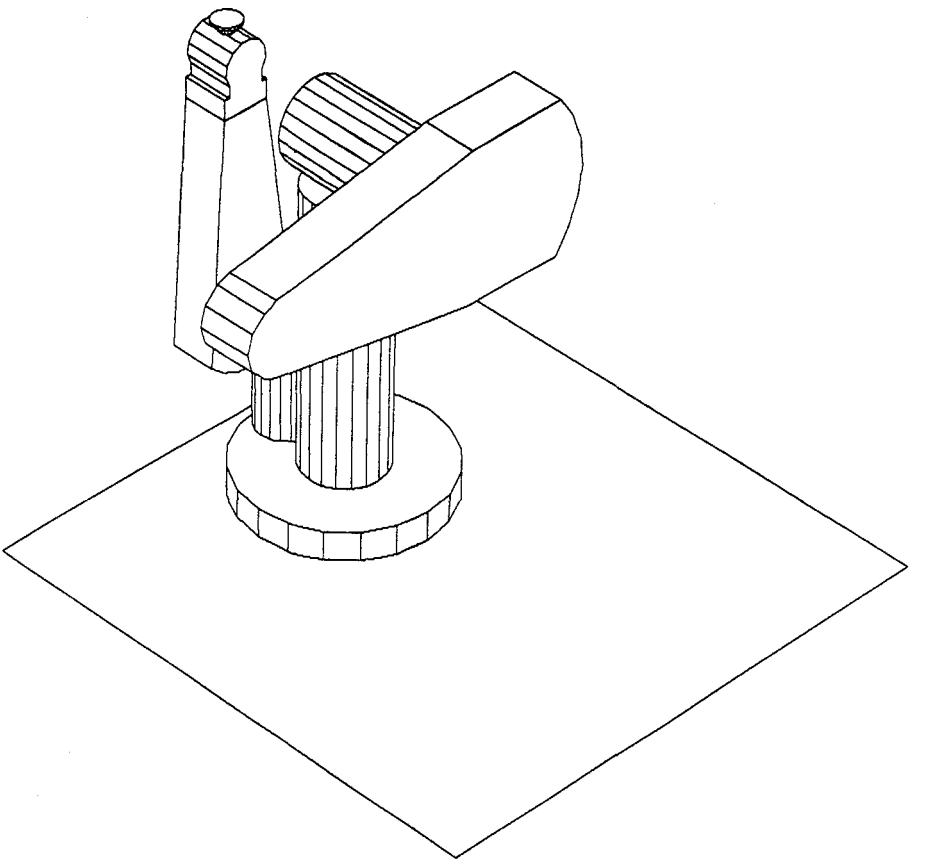


Figure 6.1. PUMA 560 robot.

should be located with only the constraints of the working environment in mind. In this example, the measurement device is a small coordinate measuring machine (CMM) located along one boundary of the workspace. The CMM uses a reference cube to define the origin of the workspace and, for consistency, the workspace frame is located so that its origin is defined by the reference cube and the coordinate axes are aligned with the axes of the CMM. A reading from the CMM, therefore, locates a point in the workspace coordinate system. This frame will be referred to as frame B and is illustrated in Figure 6.2. Frames 0 through 5 are assigned next according to the modified DH convention. The only parallel axes are joint axes 2 and 3. The location of all frames, therefore, is determined by the common normal between the axes as defined by the standard DH procedure except for frame 2, which is located according to the modified DH convention described in Chapter 2. As also described in Chapter 2, it is important that the end effector coordinate system be arbitrarily located. This means that six parameters must be used to relate coordinate systems 5 and 6.

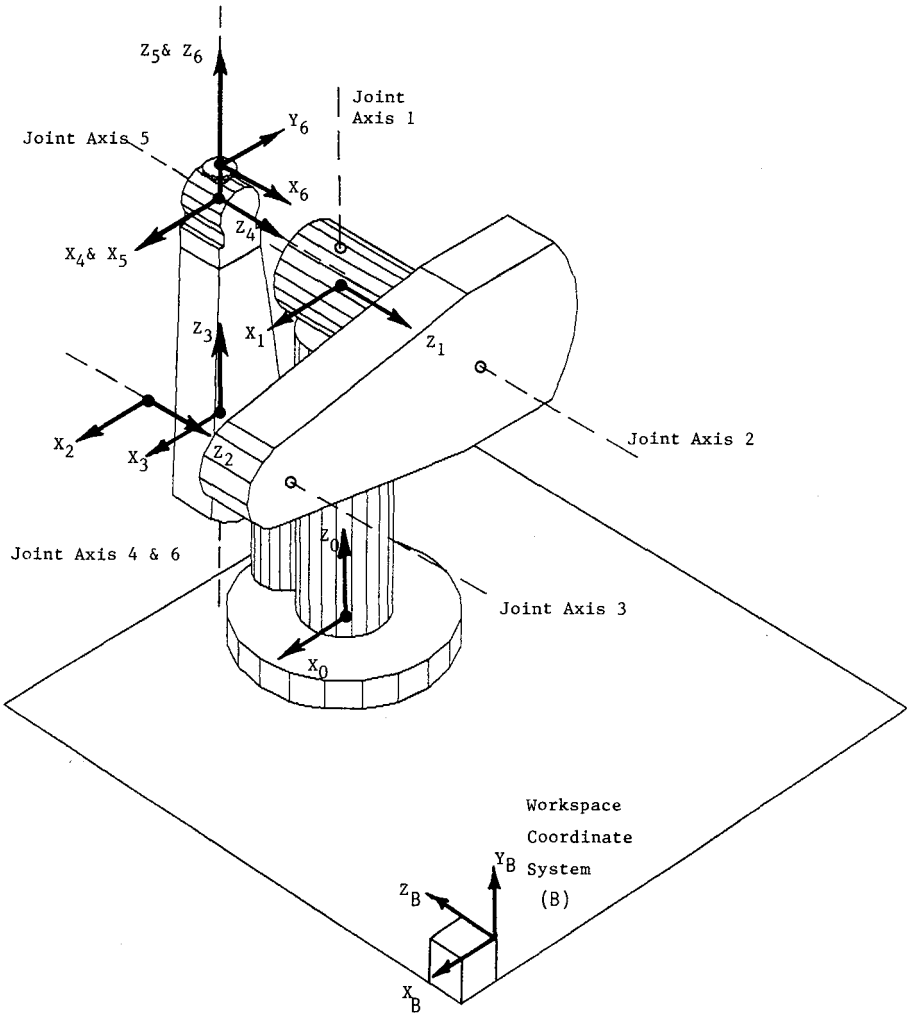


Figure 6.2. Coordinate frame assignment.

In this model, therefore, there are three distinct forms for the link transformation matrices: the standard DH, the modified DH, and a full six parameter transform. For convenience, we will refer to the standard DH as a Type 1 transformation. As described in Chapter 2, this matrix has four parameters $(r_i, l_i, \alpha_i, \theta_i)$ and is written as follows

$$\mathbf{A}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & l_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & l_i\sin\theta_i \\ 0 & \sin\alpha & \cos\alpha & r_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

TABLE 6.1 Nominal Parameters for PUMA 560

Transformation	Type	Parameter Number					
		1	2	3	4	5	6
B → 0	1	32.878	−15.591	−90.000	0.000	—	—
0 → 1	1	14.681	−0.000	−90.000	0.000	—	—
1 → 2	2	17.000	0.000	0.000	0.000	—	—
2 → 3	1	5.870	−0.800	90.000	0.000	—	—
3 → 4	1	17.050	0.000	−90.000	0.000	—	—
4 → 5	1	0.000	0.000	90.000	0.000	—	—
5 → 6	3	0.000	0.000	2.213	0.000	0.000	90.000

Since all of the joints of a PUMA 560 are revolute, the parameter θ_i is the joint variable for joint i and is specified by reading the joint transducer at each robot pose. In this example, however, we will assume that there is a possibly significant joint offset, $\delta\theta_i$, on each joint and this will be included in the model. At each pose, θ_i will be given by

$$\theta_i = \Theta_i + \delta\theta_i \tag{6.2}$$

where Θ_i is the value for the joint displacement given by the PUMA controller and $\delta\theta_i$ is the constant joint offset that is to be identified. The four variables to be determined for the Type 1 matrix, therefore, are r_i, l_i, α_i, and $\delta\theta_i$. The values of these parameters for a perfect or nominal PUMA 560 are given in Table 6.1. All of the transformations listed as Type 1 have four parameters as follows:

Parameter Number	Type 1 Parameter	Units
1	d_i	Inches
2	l_i	Inches
3	α_i	Degrees
4	$\delta\theta_i$	Degrees

As mentioned earlier, joints 2 and 3 are very nearly parallel and the modified DH convention should be applied. We will refer to this as a Type 2 transformation. As described in Chapter 2, this transformation also consists of four parameters (r_i, α_i, β_i, and θ_i). The transformation matrix is given by

$$\mathbf{A}_3 = \begin{bmatrix} -s\alpha_3 s\beta_3 s\theta_3 + c\beta_3 c\theta_3 & -c\alpha_3 s\theta_3 & s\alpha_3 c\beta_3 s\theta_3 + s\beta_3 c\theta_3 & r_n c\theta_3 \\ s\alpha_3 s\beta_3 c\theta_3 + c\beta_3 s\theta_3 & c\alpha_3 c\theta_3 & -s\alpha_3 c\beta_3 c\theta_3 + s\beta_3 s\theta_3 & r_n s\theta_3 \\ -c\alpha_3 s\beta_3 & s\alpha_3 & c\alpha_3 c\beta_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

where $s\alpha$ represents $\sin\alpha$ and so on. The variable θ_3 is the joint variable in this case also and, as before, we will assign a constant offset, $\delta\theta_3$ to be identified. The nominal values for the PUMA 560 are listed for this transformation in Table 6.1 as well. The parameters for a Type 2 transformation are as follows:

Parameter Number	Type 1 Parameter	Units
1	l_3	Inches
2	α_3	Degrees
3	β_3	Degrees
4	$\delta\theta_3$	Degrees

The final transformation relates coordinate frame 6 to coordinate frame 5. Since we do not wish to place any restrictions on the location of the end effector coordinate frame (frame 6), the transformation relating frames 5 and 6 must include six independent parameters. We will refer to this final transformation as a Type 3 transformation. This transformation is constructed from the following series of motions:

$$\mathbf{A}_6 = \mathbf{R}(z,\theta_z)\mathbf{R}(y,\theta_y)\mathbf{R}(x,\theta_x)\mathbf{T}(dx,0,0)\mathbf{T}(0,dy,0)\mathbf{T}(0,0,dz) \tag{6.4}$$

When combined, the rotations and translations expressed in Equation 6.4 may be expressed as

$$\mathbf{A}_6 = \begin{bmatrix} c\theta_y c\theta_z & -c\theta_x s\theta_z + s\theta_x s\theta_y c\theta_z & c\theta_x s\theta_y c\theta_z + s\theta_x s\theta_z & a_{14} \\ c\theta_y s\theta_z & c\theta_x c\theta_z + s\theta_x s\theta_y s\theta_z & c\theta_x s\theta_y s\theta_z - s\theta_x c\theta_z & a_{24} \\ -s\theta_y & s\theta_x c\theta_y & c\theta_x c\theta_y & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.5}$$

where

$$a_{14} = dzc\theta_x s\theta_y c\theta_z + dzs\theta_x s\theta_z - dyc\theta_x s\theta_z + dys\theta_x s\theta_y c\theta_z + dxc\theta_y c\theta_z \tag{6.6}$$

$$a_{24} = dzc\theta_x s\theta_y s\theta_z - dzs\theta_x c\theta_z + dyc\theta_x c\theta_z + dys\theta_x s\theta_y s\theta_z + dxc\theta_y s\theta_z \tag{6.7}$$

$$a_{34} = dzc\theta_x c\theta_y + dys\theta_x c\theta_y - dxs\theta_y \tag{6.8}$$

As indicated above, there are six parameters in the final transformation. Three of the parameters, θ_x, θ_y, and θ_z, describe the relative orientation of the two frames and three parameters, dx, dy, and dz, describe the position of the origin of frame 6 with respect to frame 5. The nominal values of these parameters of the PUMA 560 are given in Table 6.1. The parameter numbers are as follows.

Parameter Number	Type 1 Parameter	Units
1	dx	Inches
2	dy	Inches
3	dz	Inches
4	θ_x	Degrees
5	θ_y	Degrees
6	θ_z	Degrees

This completes the modified DH model for the PUMA manipulator. As demonstrated in Table 6.1, the model has a total of 30 parameters. According to the formula for completeness given in Chapter 2, this represents a complete model and since we have used the modified DH approach, the model will exhibit proportionality. It is important to note that this model does not include any nongeometric effects such as link deflection or gear backlash. If the accuracy enhancement provided by the model described above is not sufficient for the intended application, it may be desirable to add some nongeometric components to the model and attempt the calibration again.

6.1.2 Zero Reference Position Model

An alternative approach to modeling the PUMA manipulator is to use the zero reference position method. As described in Chapter 2, this approach consists of identifying a unit vector, $\mathbf{u}_i$, and a locating point, $\mathbf{p}_i$, for each joint axis, i. This procedure is begun by placing the robot in the configuration in which each of the joint displacements is zero. This "zero position" will serve as the reference position from which all of the robot motions will be measured. Figure 6.3 is an illustration of the PUMA manipulator in the zero position. While the zero position may be defined arbitrarily, it is convenient in this example to use the same zero position as defined by the DH procedure in the preceeding section.

Once a zero position has been determined, the workspace coordinate system must be defined. The position and orientation of this coordinate system are arbitrary and are usually located so as to meet the needs of the robot task. As described in the previous section, the CMM that will be used for data acquisition uses a reference cube to define the origin of its coordinate system. We will, therefore, define the CMM coordinate system to also be the workspace coordinate system. The location of the workspace coordinate system is illustrated in Figure 6.3.

The vectors $\mathbf{u}_i$ and $\mathbf{p}_i$ may now be defined. Since the joint axes of the perfect or nominal robot are aligned with the axes of the workspace coordinate system when the robot is in the zero position, the values for $\mathbf{u}_i$ and $\mathbf{p}_i$ are easily specificed. Table 6.2 lists the values of $\mathbf{u}_i$ and $\mathbf{p}_i$ for the nominal robot. In the table, the vector $\mathbf{p}$ is given in inches and $\mathbf{u}$ is dimensionless.

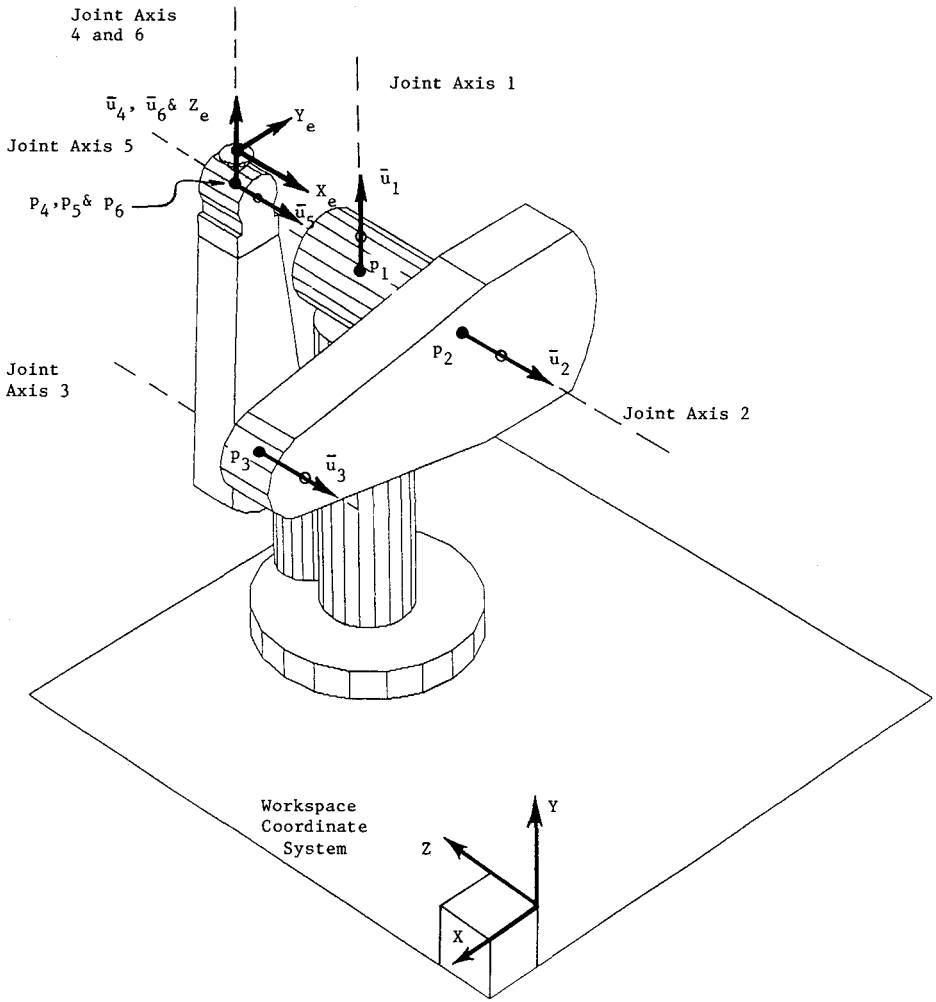


Figure 6.3. Zero reference position method.

TABLE 6.2. Zero Reference Position Parameters for Nominal PUMA

Joint	u_x	u_y	u_z	p_x	p_y	p_z
1	0	1	0	−15.00	14.70	33.60
2	0	0	−1	−15.00	14.70	27.73
3	0	0	−1	2.00	14.70	27.73
4	0	1	0	1.20	31.75	27.73
5	0	0	−1	1.20	31.75	27.73
6	0	1	0	1.20	31.75	27.73

Since we are interested in variations of the actual robot from the nominal, a set of variables must now be defined that reflect these variations. Variations in the points $\mathbf{p}_i$ that locate the joint axes in space are described by adding small displacements in orthogonal directions that are parallel to the axis of motion. For example, the axis of joint 1 is roughly parallel to the Y coordinate axis. Variations in location of this axis may be described by adding a quantity, p_x, in the X direction and a quantity, p_z, in the Z direction. The vector $\mathbf{p}_1$ may then be expressed as

$$\mathbf{p}_1 = (p_{1x} - 15.00)\mathbf{i} + 14.70\mathbf{j} + (p_{1z} + 33.60)\mathbf{k} \tag{6.9}$$

where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are unit vectors in the X, Y, and Z directions. Variations in the orientation of $\mathbf{u}$ may be expressed by specifying components of the vector along the coordinates axes that are perpendicular to the nominal vector. Using this approach, $\mathbf{u}_1$ would be expressed as

$$\mathbf{u}_1 = u_{1x}\mathbf{i} + \sqrt{1 - u_{1x}^2 - u_{1z}^2}\,\mathbf{j} + u_{1z}\mathbf{k} \tag{6.10}$$

Extending this approach to all of the joint axes, the zero reference position parameters for the actual robot may be expressed in terms of a set of unknown values that is to be determined through the calibration process. These parameters are given in Table 6.3.

As indicated in Table 6.3, there are 24 parameters associated with the vectors $\mathbf{u}_i$ and $\mathbf{p}_i$. Since a complete model for the PUMA will contain 30 parameters, we must still identify six model parameters. As discussed in Chapter 2, these parameters are associated with the zero position of the manipulator. We have the choice of establishing the desired zero position by specifying the $\mathbf{T}_0$ matrix and choosing six joint offsets as the additional unknowns or letting the six parameters that define $\mathbf{T}_0$ be the unknowns and have no joint offsets. As was stated in earlier chapters, one of the goals of the calibration process is to ensure that the zero position is independent of robot geometry and the same for all robots of a given type. For this reason, we will choose to assign $\mathbf{T}_0$ and let the additional six unknowns be the joint offsets. In this example the value chosen for $\mathbf{T}_0$ is

$$\mathbf{T}_0 = \begin{bmatrix} 0 & -1 & 0 & 1.2 \\ 0 & 0 & 1 & 33.96 \\ -1 & 0 & 0 & 27.73 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

where the upper left 3×3 partition indicates the desired orientation of the end effector coordinate system in the zero position and the fourth column represents the location of the origin. The 30 model parameters, therefore, are the 12 elements of the vectors $\mathbf{u}_i$, 12 elements of the vector $\mathbf{p}_i$, and the 6 joint offests as given in Equation 6.2.

TABLE 6.3. Zero Reference Position Parameters for Actual PUMA

Joint	u_x	u_y	u_z	p_x	p_y	p_z
1	u_{1x}	$\sqrt{1-u_{2x}^2-u_{1z}^2}$	u_{1z}	$p_{1x}-15.00$	14.70	$p_{1z}+33.60$
2	u_{2x}	u_{2y}	$-\sqrt{1-u_{2x}^2-u_{2y}^2}$	$p_{2x}-15.00$	$p_{2y}+14.70$	27.73
3	u_{3x}	u_{3y}	$-\sqrt{1-u_{3x}^2-u_{3y}^2}$	$p_{3x}+2.00$	$p_{3y}+14.70$	27.73
4	u_{4x}	$\sqrt{1-u_{4x}^2-u_{4z}^2}$	u_{4z}	$p_{4x}+1.20$	31.75	$p_{4z}+27.73$
5	u_{5x}	u_{5y}	$-\sqrt{1-u_{5x}^2-u_{5y}^2}$	$p_{5x}+1.20$	$p_{5y}+31.75$	27.73
6	u_{6x}	$\sqrt{1-u_{6x}^2-u_{6z}^2}$	u_{6z}	$p_{6x}+1.20$	31.75	$p_{6z}+27.73$

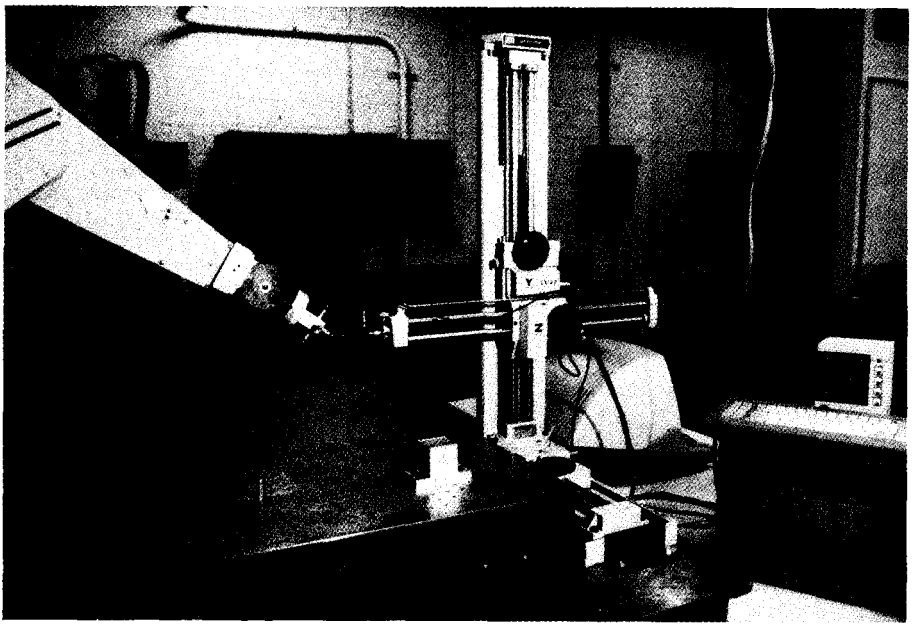


Figure 6.4. Experimental setup for measurement and evaluation.

6.2 MEASUREMENT

The experimental set-up consists of a PUMA 560 robot and a Mitutoyo Model CX-D2 coordinate measuring machine (CMM) as illustrated in Figure 6.4. The working volume of the CMM is a parallelpiped measuring 400 mm in width, 500 mm in length, and 800 mm in height. Clearly, this does not cover the entire workspace of the PUMA manipulator and all measurements are restricted to the intersection of the two working volumes. The CMM has a repeatability of 0.01 mm and a published accuracy of approximately 0.1 mm. Measurements are made by manually moving the CMM one axis at a time until the touch probe mounted on the CMM contacts an object to be measured. When the touch probe is triggered, the X, Y, and Z coordinates of the probe are recorded on the CMM display and stored in an internal buffer so that they may be transferred to a recording device.

An IBM Industrial Personal Computer is used to record the manipulator configuration, collect the CMM data, and perform some data reduction. Figure 6.5 illustrates the relationship of the various devices that comprise the experimental set-up. In addition to the active devices, several fixtures are used in the measurement process. The origin of the workspace coordinate system is defined by one corner of a cube that is securely fixed in the working volume of the CMM. The orientation of the workspace coordinate system is defined by the orientation of the axes of the CMM. The CMM has a setting which allows the "zero" position

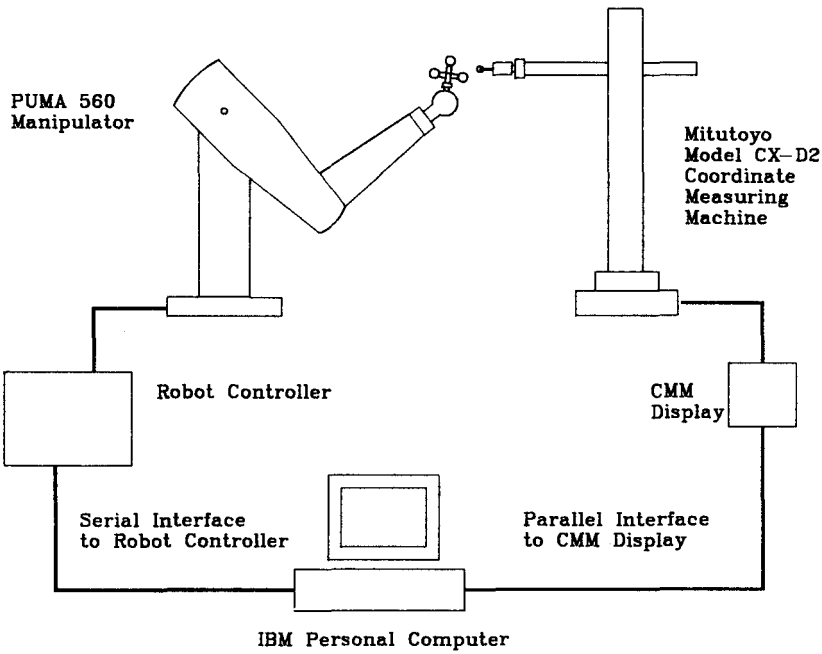


Figure 6.5. Schematic of measurement equipment.

of any axis to be established when the touch probe is triggered. The origin of the workspace coordinate system is established by successively touching the three faces of the cube at the beginning of each data collection session. Several experiments to test the repeatability of this approach were conducted. Repeated measurements of the "zero" position were made over a period of several hours and the origin was found to vary no more than one resolution unit (0.01 mm) in any reading and to have a mean value of 0.00 mm. The other fixture used in the measurement process is the end effector, which is illustrated in Figure 6.6. As shown in the figure, the end effector consists of five tooling balls roughly positioned along the coordinates axes of the end effector coordinate system. The end effector is equipped with five tooling balls so that at least three balls will always be accessible to the CMM for any end effector pose that is within the working volume of the CMM. After construction, the end effector was calibrated by using the CMM to accurately determine the location of the center of each tooling ball in the end effector coordinate system. In addition, a face plate for the manipulator was designed with two dowel pins to allow removal and replacement of the end effector without significantly changing the relationship between the end effector and the robot. It is important to note that such a repeatable and well-characterized tooling interface is vital if the results of a calibration are to be useful for a number of different end effectors.

Given the various components described above, measurement of a robot pose

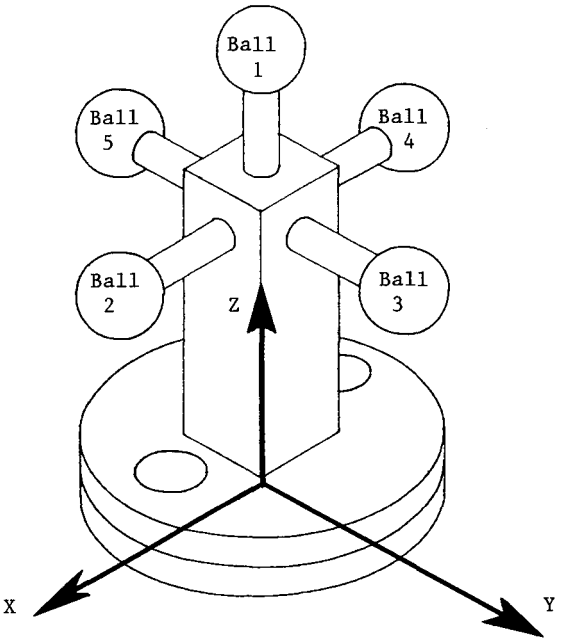


Figure 6.6. End effector.

is accomplished in the following manner. At the beginning of the measurement process, the PC prompts the operator to "zero" the CMM and asks the number of measurements to be used on each tooling ball. Since the location of the tooling ball center must be computed, a number of measurements on the surface of the ball must be made to determine the location of the ball center. Obviously, a minimum of three measurements must be made. More measurements, however, will tend to minimize the influence of noise on the estimated ball position. Using a least squares approach, the position of the ball center can be determined for as many as 10 measurements. For the data reported here, four measurements were used to locate each ball center. Once the number of measurements per ball has been entered, the measurement process is begun. The PC prompts the operator to move the robot to a new pose. When this is accomplished, the robot joint angles are automatically recorded in the PC and the operator is asked which ball will be measured next. If the operator wishes to begin with ball 2, he enters this number and then moves the CMM so that the touch probe triggers on four points on the surface of ball 2. The location of the center of ball 2 in the workspace coordinate system is then estimated and stored as $\mathbf{x}_{b2w}$ where the w subscript denotes the coordinate frame and the b2 represents the ball number. The process is continued until three balls have been measured. This information is then used

to determine the end effector pose as given by the transformation matrix $\mathbf{T}_{wt}$. We will use the symbol $\mathbf{T}_{ba}$ to denote the homogeneous transformation from coordinate system a to coordinate system b. The following relationship may be written for each ball measured.

$$\mathbf{x}_{biw} = \mathbf{T}_{we}\mathbf{x}_{bie} \tag{6.12}$$

where the subscript e denotes the end effector coordinate system. Recall that the values of $\mathbf{x}_{bie}$ were recorded during the end effector calibration. As an example, assume that measurements were made for balls 2, 3, and 5. The location of a fictional fourth ball may be defined as

$$\mathbf{x}_f = (\mathbf{x}_{b3} - \mathbf{x}_{b2}) \times (\mathbf{x}_{b5} - \mathbf{x}_{b2}) + \mathbf{x}_{b2} \tag{6.13}$$

Since this definition of $\mathbf{x}_f$ will be valid in any coordinate system, the following relationship may be written.

$$[\mathbf{x}_{b2w}, \mathbf{x}_{b3w}, \mathbf{x}_{b5w}, \mathbf{x}_{bfw}] = \mathbf{T}_{we}[\mathbf{x}_{b2e}, \mathbf{x}_{b3e}, \mathbf{x}_{b5e}, \mathbf{x}_{bfe}] \tag{6.14}$$

which yields

$$\mathbf{T}_{we} = [\mathbf{x}_{b2w}, \mathbf{x}_{b3w}, \mathbf{x}_{b5w}, \mathbf{x}_{bfw}][\mathbf{x}_{b2e}, \mathbf{x}_{b3e}, \mathbf{x}_{b5e}, \mathbf{x}_{bfe}]^{-1} \tag{6.15}$$

After the end effector pose is determined from Equation 6.15, it is stored in a data file with the joint displacements. The measurement process is continued until a predetermined number of poses have been acquired.

To determine the repeatability of the pose estimation process described above, the manipulator was placed in an arbitrary pose in the approximate center of the CMM workspace. The pose was measured 50 times and variations of the estimated pose were examined. For each measured pose, the following matrix was determined

$$\Delta_{\text{Test}} = \mathbf{T}_{we1}^{-1}\mathbf{T}_{wei} \tag{6.16}$$

where $\mathbf{T}_{we1}$ is the first measured pose and $\mathbf{T}_{wei}$ is the ith measured pose. The matrix Δ_{Test} represents the small variation between the ith and the first pose. Treating Δ_{Test} as a differential transformation, the following characterization is valid

$$\Delta_{\text{Test}} = \begin{bmatrix} 1 & -\delta_z & \delta_y & d_x \\ \delta_z & 1 & -\delta_x & d_y \\ -\delta_y & \delta_x & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.17}$$

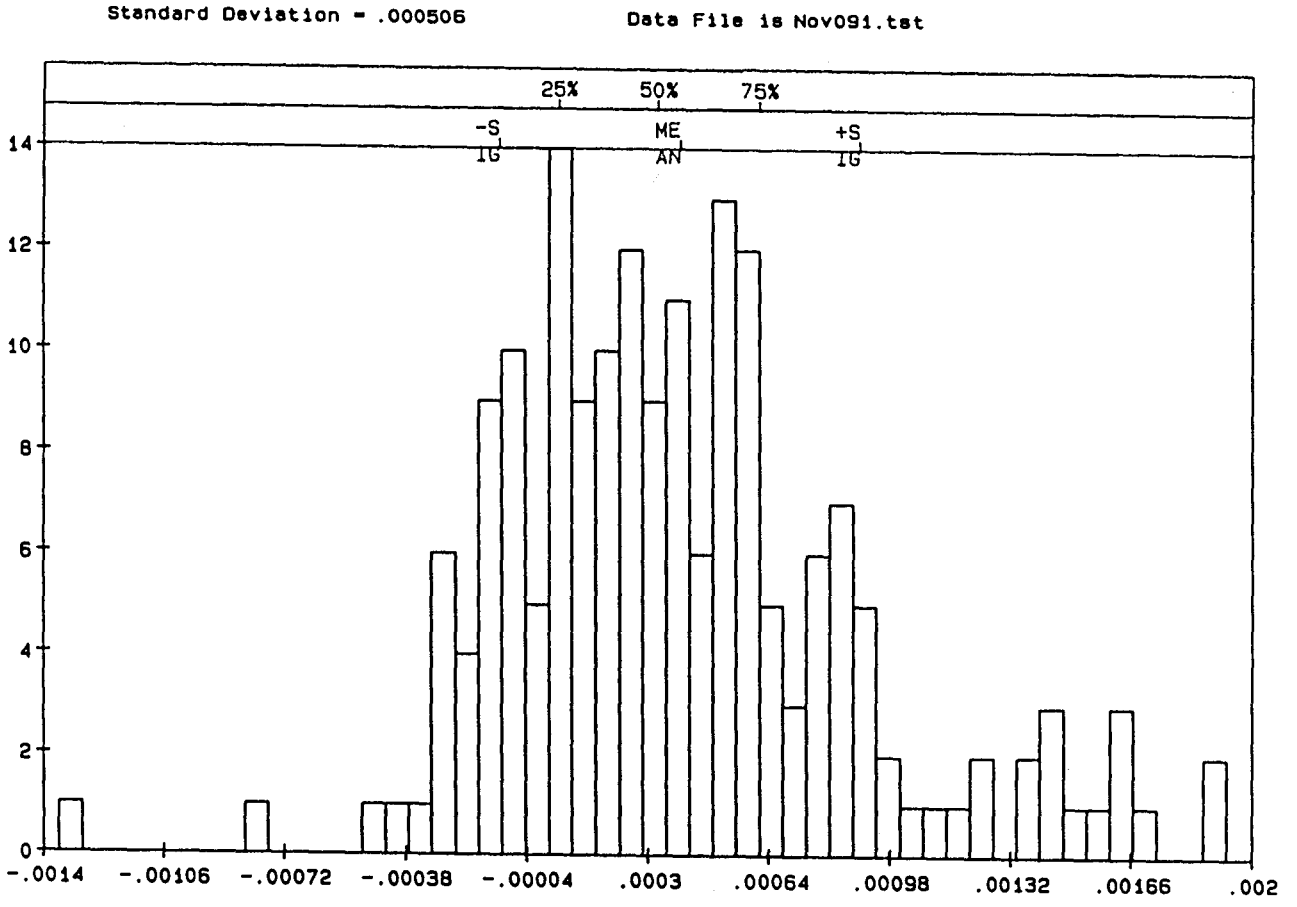


Figure 6.7. Distribution of values of δx.

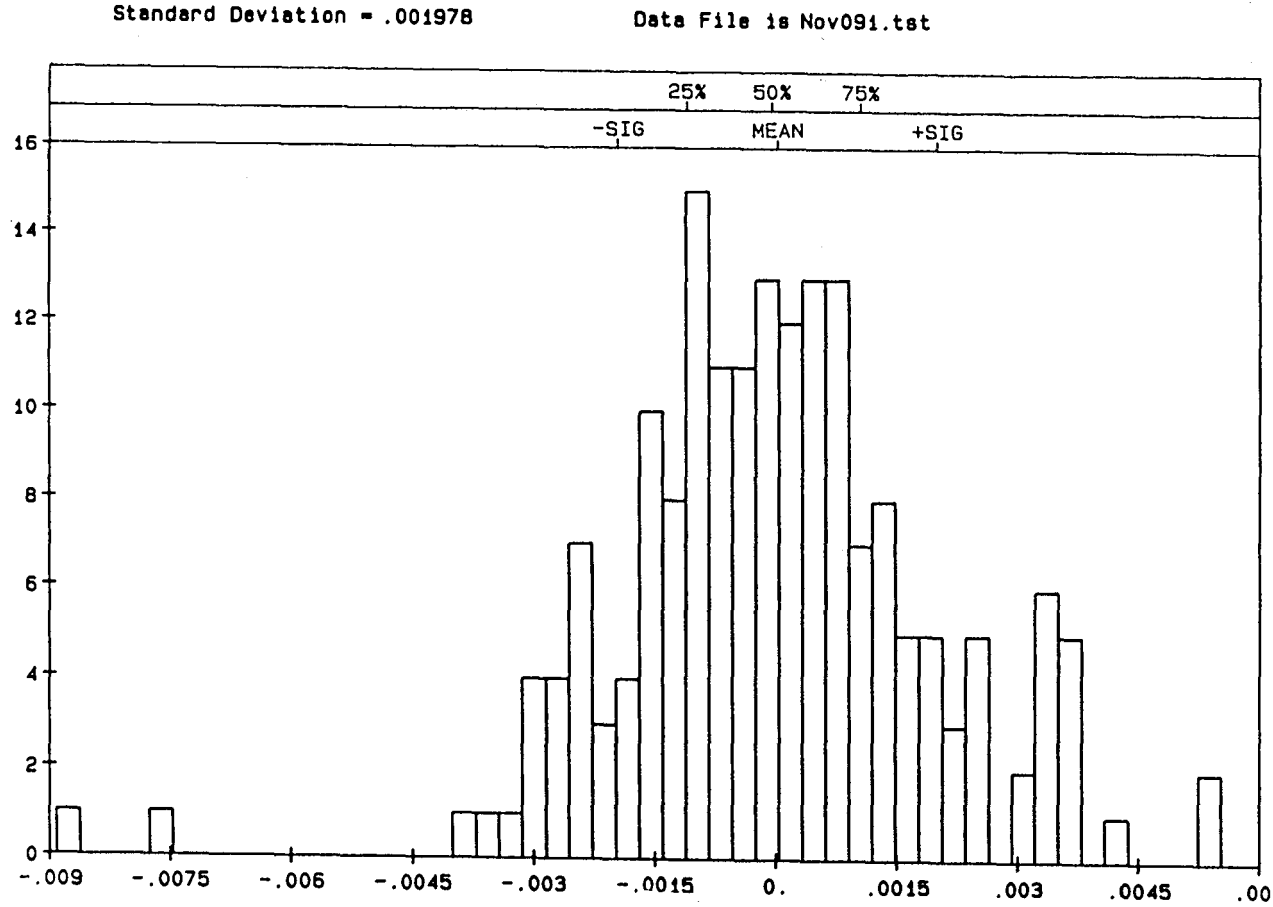


Figure 6.8. Distribution of values of dx.

where the δ_i values represent differential rotations about the coordinate axes and the d_i values are small displacements along the coordinate axes. Since the first pose was chosen as the reference values, the mean of the distribution of the differential motions should not be meaningful, but the variance indicates the repeatability of the pose estimation procedure. Figures 6.7 and 6.8 show the distributions of δ_x and d_x. Aside from the mean values, the forms of the distribution for all of the rotation parameters were similar as were the distributions for the displacements. Table 6.4 gives the standard deviation for each of the variables. As shown in the table, the measurement system has a position repeatability with a standard deviation of about 0.05 mm in every direction and an orientation repeatability with a standard deviation of 0.0006 mm in each direction. Given the repeatability of the robot and the design of the CMM, these numbers are quite acceptable.

While repeatability of the measurement system is important, the key performance measure is accuracy. Unfortunately, in this case study, no more precise system was available with which to study the accuracy of the CMM. The published accuracy of the system is 0.1 mm over the entire working volume, which is suitable for the calibration model proposed.

Using the measurement system described above, two data sets containing 50 poses each were collected. The poses were collected by roughly positioning the origin of the end effector coordinate system on the points of a $3 \times 2 \times 2$ grid in the workspace of the CMM. The 12 points of this grid are illustrated in Figure 6.9. At each point, pose measurements were taken with the manipulator in the "left arm," "right arm," "elbow up," and "elbow down" configurations as illustrated in Figure 6.10. The orientation of the end effector was varied randomly by the operator so that a range of different configurations was measured in each data set. The data were acquired in this manner so as to ensure that each joint moved through as large a range as possible during the measurement process. Since we are attempting to identify the position and orientation of the revolute axes, each joint must be rotated through a range that is large enough to ensure proper identification of the axis. Assume, for example, that a revolute joint undergoes three rotations of 5° and that the end effector location is measured after each rotation. As is illustrated in Figure 6.11, the measured points are nearly colinear and any small measurement error can significantly effect the estimated position of the rotation axis. If, however, each rotation is 120°, the measured points form a circle and measurement errors have much less effect.

The two data sets were collected by manually moving the CMM to measure the end effector locations and about 4 hr were required to acquire one set of 50 poses. Although the data proved to be suitable for robot calibration, the manual approach to data acquisition was tedious and time consuming. An automated approach based on one of the systems described in Chapter 3 would be much more desirable in a production rather than a laboratory environment.

TABLE 6.4. Estimated Standard Deviations of the Measurement Errors

Variable	Standard Deviation
δ_x	0.000506 (radians)
δ_y	0.000592 (radians)
δ_z	0.000621 (radians)
d_x	0.05024 (mm)
d_y	0.04364 (mm)
d_y	0.05654 (mm)

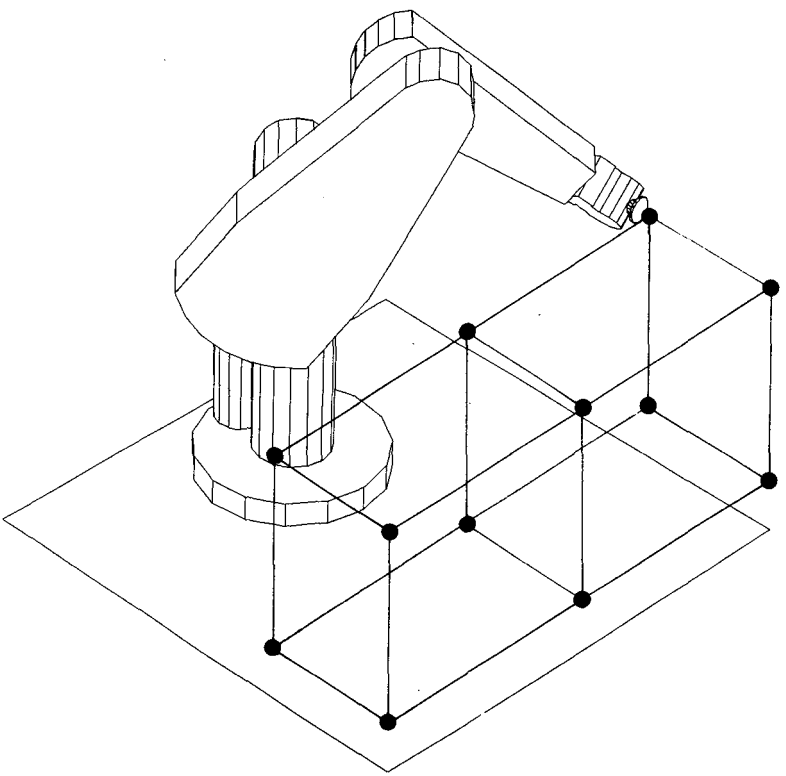


Figure 6.9. Measurement grid for data acquisition.

6.3 IDENTIFICATION

Having developed two suitable models and acquired several data sets, we may now proceed to the identification of the model parameters. Although most of the techniques described in Chapter 4 would be suitable for this process, a gradient based Levenberg–Marquardt algorithm (Section 4.4.2) was chosen. This particu-

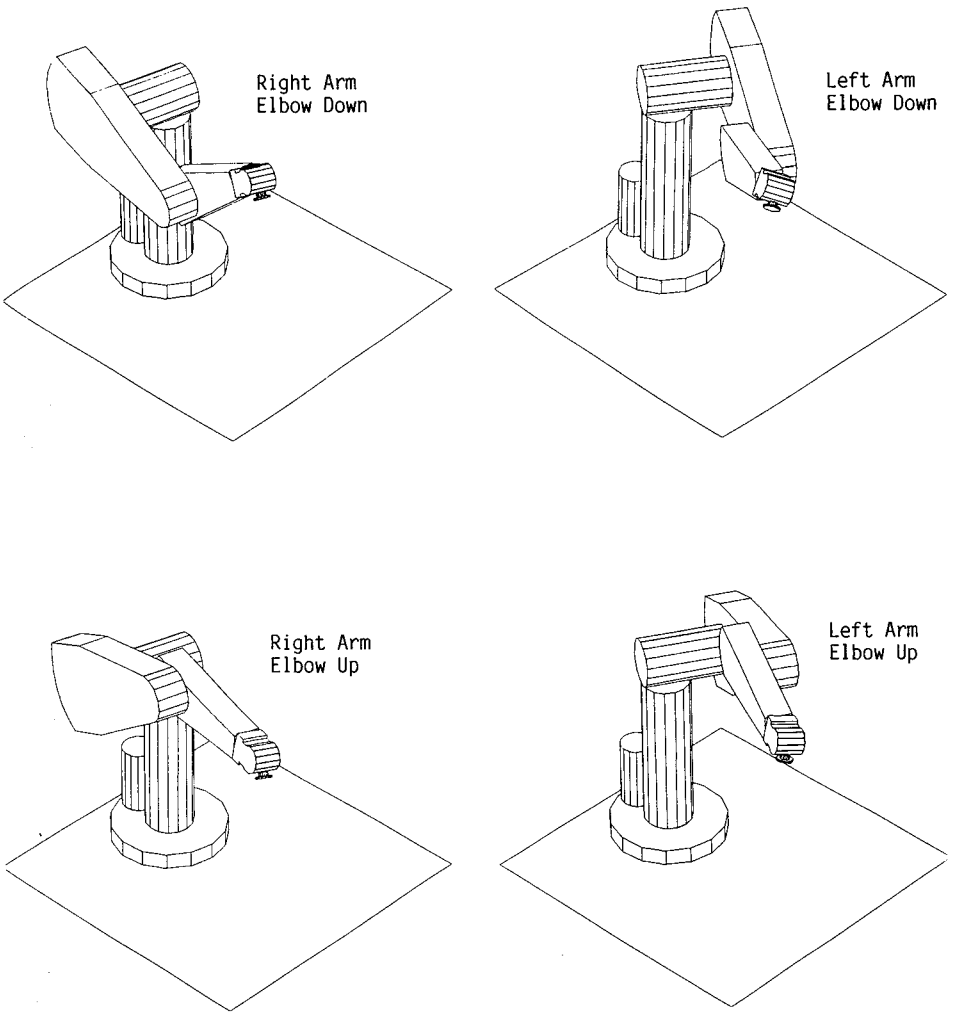
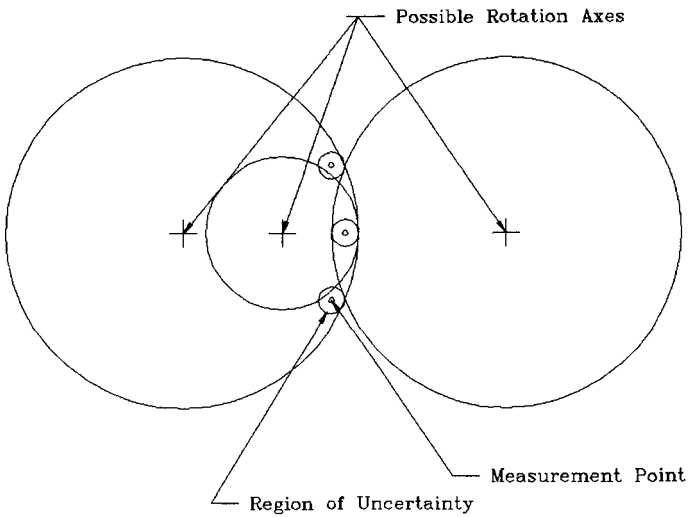
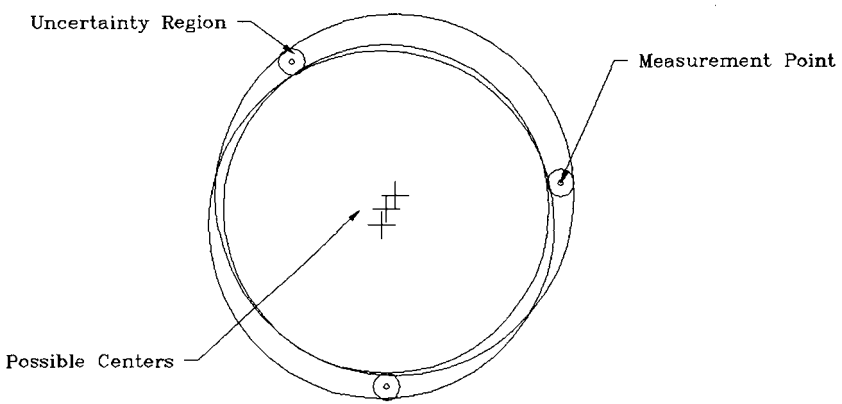


Figure 6.10. Measurement configurations.

lar algorithm was chosen primarily on the basis of ease of use. As described in Chapter 4, a FORTRAN subroutine that executes this algorithm is available in the IMSL Library. To implement this algorithm, therefore, one needs only to develop an associated subroutine that computes a performance index based on the current estimate of the model parameters. The ISML routine (ZXSSQ) numerically estimates a gradient and generates an improved estimate of the parameters until a convergence criterion is reached. Since there are 30 model parameters, the estimation of the gradient causes this algorithm to be quite slow. Since deviation of an analytical gradient for either model is a tedious and time-consuming process, it was decided to use the numerical procedure. Again,



a) Nearly Colinear Points



b) Noncolinear Points

Figure 6.11. Effect of measurement error on axis location.

since speed is not of paramount importance in a laboratory environment, it was decided that the numerical approach would suffice. If a calibration procedure is intended to be part of a manufacturing process or the final phase of robot construction, a faster method for both measurement and identification must be used. The measurement process may be speeded up by use of an automated and, hence, more expensive data acquisition system. The identification algorithm may also be enhanced by including an analytical expression for the identification Jacobian or the gradient. As mentioned above, derivation of the Jacobian or gradient is a time-consuming process. If, however, the application needs dictate a fast identification step, this price must be paid.

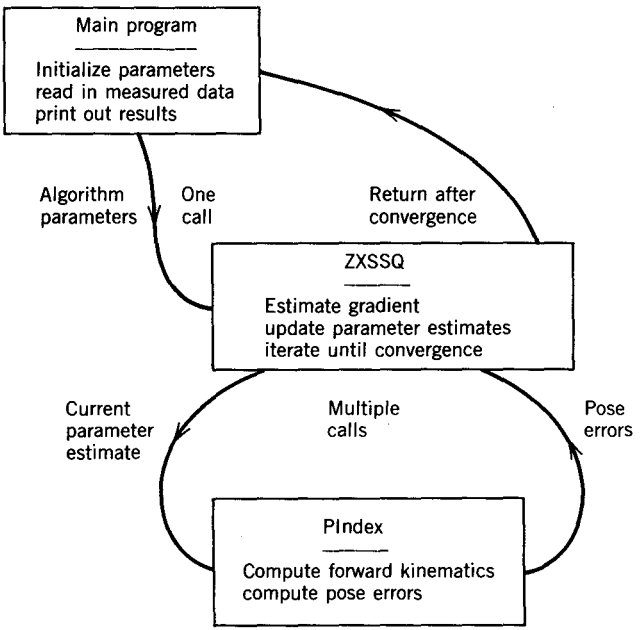


Figure 6.12. Identification algorithm.

6.3.1 Identification Algorithm

A schematic of the identification algorithm is given in Figure 6.12. As shown in the figure, the algorithm consists of three main components. The first component is the entry point in the program where the measured data are read in, the initial values of the model are defined, and the various parameters required by the identification subroutine are initialized. The second primary component is the identification subroutine ZXSSQ. This subroutine iteratively estimates the gradient and uses it to produce an updated approximation of the model parameters. The process is continued until one of several convergence criteria is met. The final component is the subroutine that takes the current estimate of the model parameters and computes an error between the model prediction and the measured data. This error is used by ZXSSQ to determine both the gradient and a performance index. Each of these components will be discussed in more detail in the following paragraphs.

The entry point or "main program" for the identification step serves to initialize the necessary variables. Figure 6.13 is a more detailed flowchart of this part of the program and a complete source listing is given in Appendix A. As illustrated in the figure, the first action is to open the measurement data file and read in the poses and associated joint displacements. Next, the parameters that are passed to subroutine ZXSSQ are initalized. This includes the vector of changes to the model parameters. Since we wish to use the nominal model as the initial guess, the vector, $\mathbf{x}$, containing the changes to the model parameters is

opportunity to converge, this is set to 1000 iterations. The argument **iopt** specifies the particular version of the algorithm that is to be used. This is set to 0 indicating Brown's algorithm without strict descent. The parameter **parm** is a vector used if **iopt** is set to 2. In this example, therefore, all four elements of **parm** are set to 0. The vector **x** has 30 elements that are initially set to 0. On return, this vector contains the modifications to the model coefficients. The argument **ssq** is returned with the sum of the squared errors of all the poses using the model coefficients that caused the algorithm to meet the convergence criterion. The parameter **f** is a vector with **m** (300) elements that is returned with the errors between the predicted and measured pose components computed with the final values of the model coefficients. The **m** (300) by **n** (30) array **xjac** is returned with an estimate of the Jacobian. The arguments **ixjac**, **xjtj**, and **work** indicate the specified dimension of **xjac** and provide work areas for the subroutine. The final two arguments, **infer** and **ier**, return indications as to which convergence criterion was used and indicate any errors that may have occured.

The final component of the identification algorithm is the subroutine that takes the current estimates of the model parameters and determines the errors between the predicted and the measured poses. In this example, the subroutine is named **PIndex**. A flowchart of **PIndex** is given in Figure 6.14. The subroutine receives the parameters **x**, **m**, and **n** from ZXSSQ and returns the vector **f**. **f** is the vector of errors between the current pose estimates and the measured poses. As described above, each of the 50 poses will contain three position and three orientation errors. The dimension of **f**, therefore, is 300. The first action in this subroutine is to use the current model coefficient modifications stored in the vector **x** to update the model parameters. This is accomplished by calling subroutine **Par**. This subroutine uses the information in **x** to modify the matrix **Param** that contains the current estimate of the model parameters. Since we have two different models, there are two different implementations of **Par** and **Param**. Both of these are given in Appendix A. Once the coefficients have been updated, a loop is entered where the current model parameters are used to estimate the pose for each data collection configuration. This is accomplished by calling **Forward**, which computes the forward kinematic model. Again, there are two versions of **Forward**, one for each model being used. The subroutine **Forward** returns a homogeneous transformation matrix, **Tr**, which describes the estimated pose. This matrix is inverted and multiplied by the measured pose, **Tm**, to obtain a differential error matrix, **Delta**. The six components of **Delta** representing the position and orientation error are added to the total error vector, **f**, and the loop is continued. Since the error between the estimated and measured pose should be small, the product of the inverse of the estimated pose and the measured pose should have the following form:

$$\mathbf{Tr}^{-1}\mathbf{Tm} \approx \begin{bmatrix} 1 & -\delta z & \delta y & dx \\ \delta z & 1 & -\delta x & dy \\ -\delta y & \delta x & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.18}$$

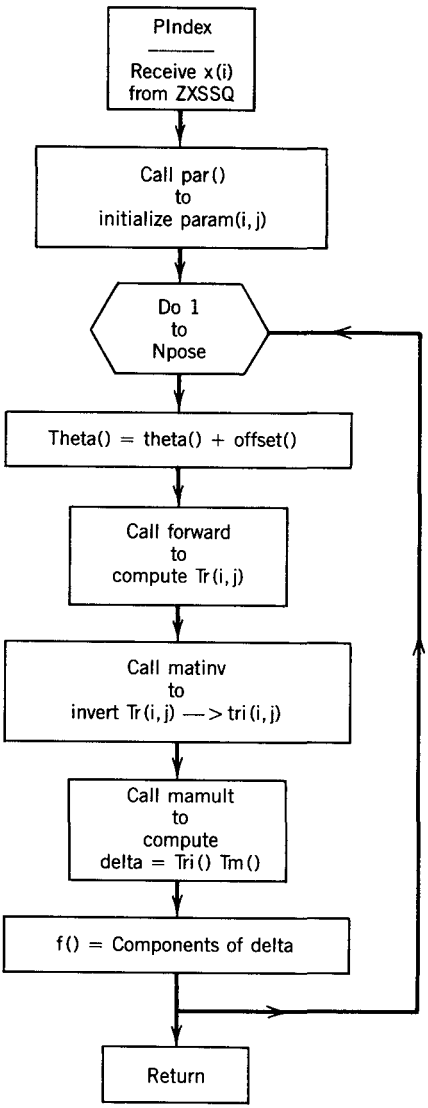


Figure 6.14. Subroutine **PIndex**.

The orientation errors, therefore, are given by elements [1, 2], [1, 3], and [2, 3] and the position errors are given by [1, 4], [2, 4], and [3, 4]. These errors will be squared so the sign is unimportant. When the loop has completed, the current coefficient vector, **x**, is printed out to indicate the progress of the algorithm.

6.3.2 Identification Results—Modified DH Model

The modified Denavit–Hartenberg model was used to compute the forward kinematics in the algorithm described above for each of the acquired data sets.

TABLE 6.5. Modified DH Parameters from Data File 1

1	33.632	−15.022	−90.121	0.086	—	—
1	14.047	−0.000	−89.981	1.962	—	—
2	17.038	0.062	−0.354	−0.109	—	—
1	5.939	−0.807	90.397	0.735	—	—
1	17.055	0.005	−90.025	−0.512	—	—
1	−0.015	−0.004	89.986	−1.443	—	—
3	−0.001	0.024	2.211	−0.019	0.140	87.901

TABLE 6.6. Modified DH Parameters from Data File 2

1	33.635	−15.021	−90.132	0.103	—	—
1	14.048	0.001	−89.974	1.959	—	—
2	17.034	0.069	−0.337	−0.098	—	—
1	5.915	−0.796	90.310	0.691	—	—
1	17.054	0.003	−89.978	−0.527	—	—
1	−0.014	−0.005	89.983	−1.442	—	—
3	−0.009	0.020	2.212	−0.006	0.208	87.907

The algorithm met the convergence criterion for each data set and the results are given in Tables 6.5 and 6.6.

As shown in the tables, each of the data sets produced approximately the same modified parameters. It is interesting to note that although there is only a slight difference between the modified parameters, there is a significant difference between the modified and the nominal parameters.

6.3.3 Identification Results—Zero Reference Position Model

The zero reference model was also used to compute the forward kinematics in the algorithm described above. The algorithm met the convergence criterion for each data set and the results are displayed in Tables 6.7 and 6.8.

The tables show that both data sets produced very similar parameter sets with one significant exception. The joint offsets for joints 4 and 6 from data set 1 are

TABLE 6.7. Zero Reference Position Parameters from Data File 1

Joint	u_x	u_y	u_z	p_x	p_y	p_z
1	−0.001	1.000	−0.002	−15.044	14.70	33.601
2	−0.003	−0.002	−1.000	−15.063	14.014	27.73
3	−0.010	−0.001	−1.000	1.926	14.690	27.73
4	0.006	1.000	0.006	1.229	31.75	27.716
5	0.407	0.002	−0.913	1.228	31.752	27.73
6	0.000	1.000	0.002	1.224	31.75	27.729

Joint	1	2	3	4	5	6
Offset	1.773	2.061	−1.889	24.054	−1.040	−26.122

TABLE 6.8. Zero Reference Position Parameters from Data File 2

Joint	u_x	u_y	u_z	p_x	p_y	p_z
1	−0.002	1.000	−0.002	−15.048	14.70	33.601
2	−0.004	−0.002	−1.000	−15.067	14.010	27.73
3	−0.010	−0.001	−1.000	1.919	14.691	27.73
4	0.006	1.000	0.004	1.224	31.75	27.707
5	0.158	0.004	−0.987	1.224	31.751	27.73
6	0.000	1.000	0.004	1.220	31.75	27.721

Joint	1	2	3	4	5	6
Offset	1.747	2.073	−1.921	9.138	−1.104	−11.209

distinctly different from those produced by data set 2. If we observe more closely, however, we see that although these offsets are different, the difference between the joint 6 offset and the joint 4 offset for each of the two data sets is approximately 2.07°. Since joint 4 and joint 6 are almost aligned in the zero position, this suggests that the joint 5 axis is rotated to a different position when the robot is at zero for each data set. This is verified by observing the axis alignment for joint 5 in the zero position as shown in Tables 6.7 and 6.8. Table 6.7 gives $\mathbf{u}_5$ as $0.407\mathbf{i} + 0.002\mathbf{j} - 0.913\mathbf{k}$. This implies that the axis is rotated approximately −24° about the Y axis, which matches the assigned joint offset. Table 6.8 gives $\mathbf{u}_5$ as $0.158\mathbf{i} + 0.004\mathbf{j} - 0.987\mathbf{j}$, which implies a −9° rotation. Again, this matches the given joint offset. The obvious question here is which set of joint offsets is correct? The answer lies in the fact that the model we used requires that the end effector have a specific pose when the robot is in the zero position. If we had a "perfect" robot with a nominal geometry, the alignment of axis 5 in the zero position could be arbitrary without affecting the end effector pose. Since the geometry of our robot is close to nominal, the actual joint offsets for joints 4 and 6 have little effect on the total accuracy as long the difference between the joint offsets is 2.07° and the zero position for axis 5 reflects the chosen offsets. This will be demonstrated in Section 6.4 where the accuracy of the identified models is compared.

6.3.4 Comparison of Identified Model Geometry

Having identified parameters for both modified DH and zero reference position models with two different data sets, it is interesting to consider the differences between the two models for a given data set. Since there are undoubtedly some unmodeled effects in the actual robot, it would seem reasonable to assume that the DH model may predict a slightly different geometry than the zero reference position model. This would imply that the DH model responds to unmodeled effects in a slightly different manner than the zero reference position model. Although it is impossible to say which might be better at this point, any differences would be interesting.

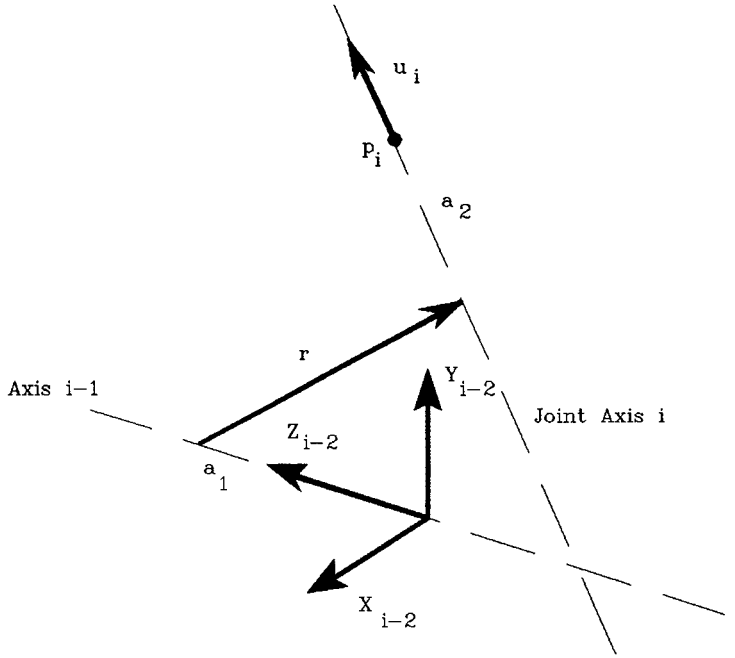


Figure 6.15. Conversion from zero reference to DH parameters.

To investigate any such differences, it is possible to use the zero reference position model parameters to compute the Denavit–Hartenberg parameters. The computed DH parameters can then be compared to the DH parameters obtained from the actual identification process and differences observed. To accomplish the conversion from zero reference position parameters, consider two consecutive joint axes, $i-1$ and i. The orientation of each of these joint axes is defined by a unit vector $\mathbf{u}_{i-1}$ and $\mathbf{u}_i$ in the zero reference model. The position is likewise defined by points $\mathbf{p}_{i-1}$ and $\mathbf{p}_i$. We will identify the DH parameters by beginning at the base coordinate system and determining the location of the common normal between the axes. This will then allow us to locate the next coordinate system according to the DH formalism and, therefore, determine the DH parameter set. To illustrate, we will assume that the DH coordinate system for axis $i-1$ has been determined. This is illustrated in Figure 6.15. Initially, the vector $\mathbf{u}_i$ and point $\mathbf{p}_i$ are defined in the base coordinate system. The first step is to use the homogeneous transformation matrix locating the axis on joint $i-1$, $\mathbf{A}_{i-2}$, to transform $\mathbf{u}_i$ and $\mathbf{p}_i$ into the axis coordinate system. This is accomplished as follows:

$$\mathbf{p}_i' = \mathbf{A}_{i-2}\mathbf{p}_i \tag{6.19}$$

$$\mathbf{u}_i' = \mathbf{R}_{i-2}\mathbf{u}_i \tag{6.20}$$

where $\mathbf{p}_i'$ and $\mathbf{u}_i'$ are the vectors after transformation and $\mathbf{R}_{i-2}$ is only the rotation component of $\mathbf{A}_{i-2}$. Having made the indicated transformations, the following expressions hold:

$$\mathbf{p}_i' = a_1\mathbf{k} + \mathbf{r} + a_2\mathbf{u}_i' \tag{6.21}$$

$$\mathbf{r}\cdot\mathbf{k} = 0 \tag{6.22}$$

$$\mathbf{r}\cdot\mathbf{u}_i' = 0 \tag{6.23}$$

where a_1, a_2, and $\mathbf{r}$ are defined in Figure 6.15. Equation 6.21 through 6.23 represent five equations in five scalar unknowns (r_x, r_y, r_z, a_1, and a_2). Equation 6.22 implies that r_z is zero. Equation 6.23 may be used to relate the remaining components of $\mathbf{r}$ as follows:

$$r_y = -\frac{u_x'}{u_y'}r_x \tag{6.24}$$

When this result is substituted into Equation 6.21, the following set of linear scalar equations results:

$$\begin{bmatrix} 1 & 0 & u_x' \\ -\frac{u_x}{u_y} & 0 & u_y' \\ 0 & 1 & u_z' \end{bmatrix}\begin{bmatrix} r_x \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} p_x' \\ p_y' \\ p_z' \end{bmatrix} \tag{6.25}$$

Solving these equations yields values for a_1, a_2, and $\mathbf{r}$. To continue, we normalize $\mathbf{r}$ to get $\mathbf{r}_n$. Now the X axis of the next DH coordinate system will be given by $\mathbf{r}_n$ and the Z axis is given by $\mathbf{u}_i'$. The Y axis, therefore, will be given by $\mathbf{u}_i' \times \mathbf{r}_n$. The next DH transformation, therefore, may be written as

$$\mathbf{A}_{i-1} = \begin{bmatrix} r_{nx} & -u_z'r_{ny} & u_x & r_x \\ r_{ny} & u_z'r_{nx} & u_y & r_y \\ 0 & u_x'r_{ny} - u_y'r_{nx} & u_z & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.26}$$

Using the various elements $\mathbf{A}_{i-1}$, the DH parameters may be written as follows:

$$\theta_{i-1} = \tan^{-1}\left(\frac{a_{21}}{a_{11}}\right) \tag{6.27}$$

$$\alpha_{i-1} = \tan^{-1}\left(\frac{a_{32}}{a_{33}}\right) \tag{6.28}$$

$$r_{i-1} = a_1 \tag{6.29}$$

$$l_{i-1} = |\mathbf{r}| \tag{6.30}$$

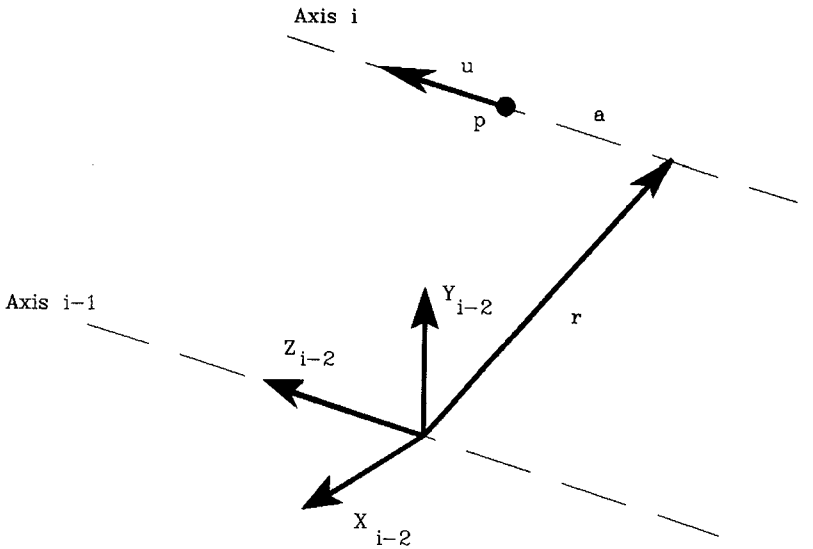


Figure 6.16. Conversion from zero reference to modified DH parameters.

The approach outlined above may be used for all axes where the standard DH model is applied. For those axes that are nearly parallel and the modified DH model is used, the following procedure may be used. As before, we begin by transforming $\mathbf{u}_i$ and $\mathbf{p}_i$ into the DH coordinate system defined for axis $i-1$. If the transformed vectors are designated as $\mathbf{u}_i'$ and $\mathbf{p}_i'$, the following equations hold:

$$\mathbf{r} + a\mathbf{u}_i' = \mathbf{p}_i' \tag{6.31}$$

$$\mathbf{r} \cdot \mathbf{k} = 0 \tag{6.32}$$

where $\mathbf{r}$ and a are as illustrated in Figure 6.16. Solving Equations 6.31 and 6.32 for $\mathbf{r}$ and a and then normalizing $\mathbf{r}$ to get $\mathbf{r}_n$ allows us to write the tranformation matrix $\mathbf{A}_{i-1}$ as follows:

$$\mathbf{a}_{i-1} = \begin{bmatrix} a_{11} & -u_z' r_{ny} & u_x' & r_x \\ a_{21} & u_z' r_{nx} & u_y' & r_y \\ a_{31} & -u_x' r_{ny} - u_y' r_{nx} & u_z' & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.33}$$

where a_{11}, a_{21}, and a_{31} are chosen so as to ensure that the matrix is orthonormal.

$$a_{11} = -a_{23}a_{32} + a_{33}a_{22} \tag{6.34}$$

$$a_{21} = -a_{12}a_{33} + a_{13}a_{32} \tag{6.35}$$

$$a_{31} = -a_{13}a_{22} + a_{12}a_{23} \tag{6.36}$$

TABLE 6.9. Comparison of DH Parameters for Data File 1

	Converted DH Parameters			Identifed DH Parameters		
Joint (Type)	1	2	3	1	2	3
1(1)	33.632	−15.022	−90.120	33.632	−15.022	−90.121
2(1)	14.048	0.000	−89.983	14.047	0.000	−89.981
3(2)	17.038	0.066	−0.353	17.038	0.062	−0.354
4(1)	5.939	−0.807	90.393	5.939	−0.807	90.397
5(1)	17.060	0.005	−90.025	17.055	0.005	−90.025
6(1)	−0.015	−0.004	−89.987	−0.015	−0.004	−89.986

The modified DH parameters may now be written as

$$l_{i-1} = |\mathbf{r}| \tag{6.37}$$

$$\alpha_{i-1} = \sin^{-1}(a_{32}) \tag{6.38}$$

$$\beta_{i-1} = \tan^{-1}\left(-\frac{a_{31}}{a_{33}}\right) \tag{6.39}$$

$$\theta_{i-1} = \tan^{-1}\left(\frac{a_{24}}{a_{14}}\right) \tag{6.40}$$

The algorithm described above was applied to convert the zero reference position parameters to modified DH parameters for both data sets. The results are given in Tables 6.9 and 6.10.

As shown in the tables, the converted parameters compare very closely with the identified parameters. This is important because it indicates that each model converges to the same physical geometry for each data set. In other words, both models respond the same to unmodeled effects and there are no advantages in using a particular model. It should also be noted that joint offsets were not

TABLE 6.10. Comparison of DH Parameters for Data File 2

	Converted DH Parameters			Identified DH Parameters		
Joint (Type)	1	2	3	1	2	3
1(1)	33.635	−15.021	−90.132	33.635	−15.021	−90.132
2(1)	14.048	0.001	−89.977	14.048	0.001	−89.974
3(2)	17.034	0.071	−0.336	17.034	0.069	−0.337
4(1)	5.916	−0.796	90.313	5.915	−0.796	90.310
5(1)	17.059	0.003	−89.977	17.054	0.003	−89.978
6(1)	−0.014	−0.005	−89.979	−0.014	−0.005	−89.983

compared. This is primarily because the zero reference model was constrained to have a particular zero position and the modified DH model is allowed to take an arbitrary zero position. The joint offsets, therefore, should not be expected to compare.

6.4 ASSESSMENT OF CALIBRATED ROBOT

To compare the accuracy of each of the identified models, a set of 10 poses that span the reachable volume of the CMM was chosen. For each of these poses, the end effector pose was measured with the CMM and the joint angles as given by the robot controller were recorded. Each model as well as the nominal model were then used to predict the measured pose. Errors between the predicted pose and measured pose were quantified in the following manner. The ith measured pose, $\mathbf{P}_{mi}$, is inverted and multiplied by the predicted pose, $\mathbf{P}_{pi}$. This product is assumed to have the form given in Equation 6.18. A position and orientation error is then defined as

$$\delta\theta = \sqrt{\delta x^2 + \delta y^2 + \delta z^2} \tag{6.41}$$

$$\delta r = \sqrt{dx^2 + dy^2 + dz^2} \tag{6.42}$$

There errors are used to evaluate the accuracy enhancement given by each of the identified models.

Since two data sets were used to identify the parameters for each model, the question arises as to which set of parameters should be used. For the modified DH model, the differences between the two parameter sets are small and the choice is somewhat arbitrary. It was decided to use the average of the two parameter sets to evaluate the accuracy of the model. The parameters used are listed in Table 6.11.

The zero reference parameter sets compared closely in every aspect but the joint offsets for joints 4 and 6. As above, the average of the parameters from each data set was used for all parameters except those related to the joint offsets for axes 4 and 6. The offset for joint 4 was arbitrarily chosen to be 5° and the offset for axis 6 as well as the orientation of u_5 was chosen to be consistent with this selection. The chosen parameters are listed in Table 6.12.

TABLE 6.11. Modified DH Parameters Used in Accuracy Test

1	33.634	−15.022	−90.127	0.095	—	—
1	14.048	−0.001	−89.978	1.961	—	—
2	17.036	0.066	−0.346	−0.104	—	—
1	5.927	−0.802	90.354	0.713	—	—
1	17.055	0.004	−90.002	−0.520	—	—
1	−0.015	−0.005	89.986	−1.443	—	—
3	−0.005	0.022	2.212	−0.013	0.174	87.904

TABLE 6.12. Zero Reference Position Parameters used in Accuracy Test

Joint	u_x	u_y	u_z	p_x	p_y	p_z
1	−0.002	1.000	−0.002	−15.046	14.70	33.601
2	−0.004	−0.002	−1.000	−15.065	14.012	27.73
3	−0.010	−0.001	−1.000	1.923	14.691	27.73
4	0.006	1.000	0.005	1.227	31.75	27.712
5	0.087	0.003	−0.996	1.226	31.752	27.73
6	0.000	1.000	0.003	1.222	31.75	27.725

Joint	1	2	3	4	5	6
Offset	1.760	2.067	−1.905	5.000	−1.072	−7.07

TABLE 6.13. Accuracy of Various Models

	Nominal		Modified DH		Zero Reference Position	
Pose	$\delta\theta$	δr	$\delta\theta$	δr	$\delta\theta$	δr
1	3.831	0.854	0.062	0.007	0.163	0.022
2	3.810	1.034	0.138	0.014	0.175	0.010
3	4.091	1.137	0.263	0.024	0.242	0.023
4	3.855	1.292	0.179	0.023	0.248	0.022
5	2.657	0.926	0.254	0.023	0.351	0.024
6	1.115	1.489	0.220	0.022	0.236	0.011
7	3.033	0.753	0.155	0.022	0.062	0.040
8	4.153	1.063	0.262	0.022	0.374	0.028
9	2.996	1.215	0.144	0.018	0.060	0.010
10	5.589	1.681	0.543	0.029	0.440	0.025

The nominal parameters as well as the parameters given in Tables 6.11 and 6.12 were used to predict the 10 selected poses. The results of this test are listed in Table 6.13. The values for $\delta\theta$ are in degrees and the values for δr are in inches.

6.5 CONCLUSION

The purpose of this chapter has been to demonstrate the use of some of the techniques presented in earlier chapters in this book. Two models, the modified DH and the zero reference position model, were used to model a PUMA 560 robot. A measurement system based on a small, manually operated CMM was used to acquire two sets of 50 poses. The models and data were then used in an identification procedure based on the Levenberg–Marquardt algorithm. The resulting parameter sets were compared and used in a test to demonstrate the level of accuracy enhancement.

As Table 6.13 indicates, both models represent a significant enhancement over the nominal model. Of course, the large levels of both position and orientation error in the nominal model reflect the fact that no significant effort was made to determine the exact location of the robot in the workspace coordinate system for the nominal model. Since these parameters are obtained as a part of the calibration process, there is no need to obtain a precise initial estimate of the manipulator's position and orientation. Other tests with the robot and measuring system described above (see Mooring and Padavala [1]) indicate that even with precise initial location of the robot, the nominal model still yields position errors on the order of 0.400 in. and orientation errors on the order of 2.3°. Since the largest position error indicated in Table 6.13 is 0.040 in., it is safe to say that the calibration procedure has improved the accuracy by at least a factor of 10. Since the repeatability of the manipulator is reported to be 0.005 in., it is reasonable to assume that further improvements in accuracy are possible with the addition of nongeometric parameters to the models and the use of more precise equipment for data collection.

REFERENCE

[1] B. W. Mooring and S. S. Padavala. The effect of model complexity on robot accuracy. In *Proceedings of 1989 IEEE Conference on Robotics and Automation*, pp. 593–598, IEEE, May 1989.

CHAPTER 7

PERFORMANCE EVALUATION

In the previous chapters of this book, we examined techniques to enhance the performance of a manipulator. Little attention, however, has been paid to the various means of quantifying robot performance. An understanding of performance metrics is important for the initial acquisition of a new manipulator system and for determining the suitability of an available machine for a proposed application as well as evaluating the effectiveness of a calibration procedure. In this chapter, we will review several robot performance standards that are currently available or are under development. This will be followed by discussions on several performance indices that are currently in use.

7.1 PERFORMANCE STANDARDS

Little formalism exists in the literature of robot performance standards due to the newness of the subject. Manufacturers generally provide sparse information about their products beyond a single valued quote of repeatability perhaps even stated as an estimate of accuracy. Industrial companies using large numbers of robots, including perhaps a wide range of models from different manufacturers, have established their own acceptance procedures that evaluate measures likely to be of importance to their proposed application. Examples of such evaluation tests are provided by Kochan [11] and include Ford Motor Company's Robotics and Automation Application Consulting Center (RAACC) and the Robotics Evaluation Center at the Industrial Technology Institute, Ann Arbor, Michigan.

As might be expected, the National Bureau of Standards (NBS—recently reorganized and renamed the National Institute of Standards Technology, NIST) has been active for some time in the evaluation of machine tools and robot

manipulators. Like robot calibration, the determination of robot performance requires accurate measurements of various parameters, although the data are not usually used to identify a set of model parameters, but to express errors from some desired objective as shown by Busch et al. [4]. Researchers at NIST have developed and evaluated several experimental methods directed at position measurements, one of the principal performance measures. Several of these techniques have been described in Chapter 3 of this book, but the work of Lau et al. [12] provides a comprehensive review.

A central axiom on which this book is based is that accuracy errors mainly depend on differences between the actual manipulator kinematics and the kinematics of the model used by the robot controller. Although it is plausible that static pose may be the most important metric, it is not clear that kinematic discrepancy is the sole cause of error. In this chapter we will define performance primitives that need to be considered before matching a particular manipulator with a specific task. These primitives are as follows:

- Resolution
- Repeatability
- Accuracy
- Path control
- Speed
- Payload
- Temperature sensitivity
- Compliance.

It will be noted that the list is probably not complete and that most of these measures are coupled. For example, the repeatability is a function both of speed and payload. In addition, some of these measures must be qualified by other parameters that are not in themselves performance primitives. Examples of these qualifiers are

- Approach direction
- Location in workspace where measurement was taken
- Closeness to singularities
- Control system gains and other parameters.

These qualifiers are necessary since, for example, the ability of a robot to execute accurate straight line motion will be severely degraded if the requested path passes through a kinematic singularity.

A further axiom underlying the material presented in this book has been to improve performance through deficiency compensation rather than engineering elimination. This philosophy may be applied to some of the performance measures so that, for example, inaccuracy due to temperature fluctuations may be

reduced if a suitable thermal model can be formulated and identified through experimental measurements.

Although there are no definitive formal standards available at the present time that address robot performance measures, considerable work is presently being done to establish comparative testing methods both in the United States and at the international level. In the United States, the Robotics Institute of America (RIA) began to address the standards issue in the early 1980s with the establishment of committees to review the following aspects of performance:

R15.01—Electrical Interface
R15.02—Human Interface
R15.03—Mechanical Interface
R15.04—Communications/Information
R15.05—Performance
R15.06—Safety

Prange and Peyton [21] provide an update on the work of these committees as of early 1988, although only that of R15.05 dealing with performance will be dealt with in any detail here.

The RIA R15.05 draft standard on robot performance [2] attempts to define how to compare different manipulator systems based on six criteria: accuracy, cycle time, repeatability, overshoot, settling time, and compliance. To do this, the document defines standard test paths and measurement points within the path, with alternatives for robots that cannot access the standard locations due to kinematic constraints. Performance classes dealing with nominal tests, and tests intended to optimize cycle time, repeatability, and other custom criteria are then defined and assessed using the six basic criteria. Although robot testing and calibration are different processes they share many of the requirements used to make precise measurements. In this regard the RIA standard is less specific. It does not indicate which measurement system should be used to measure the robot in the standard locations, or the relationship between the accuracy of the measuring system and the expected confidence in the results. Prior to testing the standard requires the user to "match the test equipment coordinate system with the robot base coordinate system," which as earlier chapters of this book have indicated, cannot be done reliably. The Standard does however define standard paths and locations against which to compare robots, but fails somewhat in addressing the rather more important issues of *how* to measure rather than *what* to measure.

On a worldwide basis, the International Standards Organization (ISO) based in Europe has Technical Committee 184 covering Industrial Automatic Systems with Subcommittee SC2 addressing Robots for Manufacturing Environments. In the United States, the ISO is represented by RIA and it is likely that RIA R15.05 will be the U.S. contribution toward the ISO/TC184/SC2 discussions.

In the remaining sections of this chapter the performance measures will be dealt with in turn. Methods by which each primitive may be measured and the appropriate qualifiers for the primitive will be indicated.

7.2 RESOLUTION, REPEATABILITY, AND ACCURACY

These three performance measures are grouped together because they are all measures of *static* quantities, and in many cases the same experimental apparatus may be used to assess each variable. Although these quantities have been introduced in previous chapters, it is worth representing repeatability and accuracy graphically as an indication of how an experiment might be designed to measure them. Figure 7.1 indicates a target, the center of which represents a defined (not taught) point, and the dots indicate the locations of the robot when repeatedly requested to go to the defined point and then move away. Note that repeatability and accuracy are three-dimensional vectors in the work area of the robot and that the graphs shown represent their worst values in one plane only. Figure 7.1a represents the case of high repeatability and high accuracy, Figure 7.1b represents high repeatability with low accuracy, and Figure 7.1c represents low repeatability with high accuracy.

Such tests, in which the robot places a stylus or pen on a piece of paper, are superficially appealing and would undoubtedly give some estimate of performance. Questions quickly arise, however, to cloud the value of the results. One might first ask where the target should be placed in the workspace, should the plane of the target be vertical, horizontal, or at some oblique angle, is the target always approached from the same direction or from different directions, and at what speed with what payload? More interestingly, if the robot base coordinate system is "inside" the body of the robot, how do we accurately know the

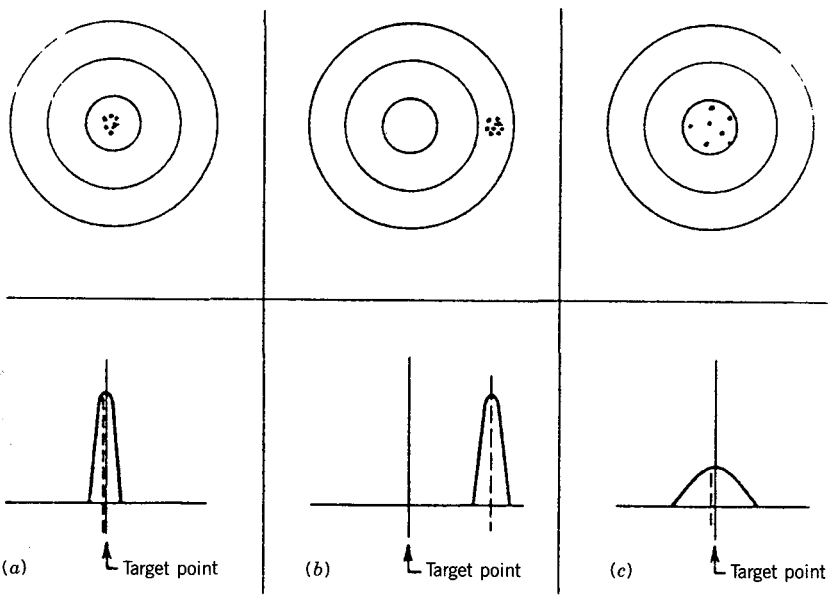


Figure 7.1. Repeatability versus accuracy. Reprinted with permission of the Society of Manufacturing Engineers, Dearborn, MI.

coordinates of the center of the target? It is not surprising that attempts to formalize the process (such as RIA R15.05) appear overly complex, and that manufacturers avoid the accuracy issue altogether and provide repeatability data only. It can be seen from Figure 7.1 that repeatability may be related to the spatial distribution of the test data on the plane, the larger the standard deviation, the poorer the repeatability. It is not clear therefore what is meant when the quoted repeatability of a particular manipulator is 0.005 in. Does it mean that 60% of the test points were within 0.005 in. of the sample set or 90% or 95% were? Such figures need clarification. Manufacturers rarely quote orientation repeatability, usually only position.

Whatever the measured repeatability, it will never be smaller than the *resolution* of the robot, which is defined as the smallest incremental move of which the machine is capable of sensing. Although this is determined solely by the individual joint servo systems that are fixed for a given machine, the resulting endpoint displacements resulting from the smallest incremental motion of any single joint will depend on the instantaneous arm configuration, and may be estimated through the manipulator Jacobian. Since resolution varies over the workspace and repeatability is a function of resolution, it is expected that repeatability will also vary throughout the workspace. This may be observed through simulation by adding random noise of a fixed Gausian or uniform distribution to each joint angle and using the forward kinematic model of the manipulator to "place" the manipulator repeatedly at various locations in the workspace. Calculating the mean and standard deviations of the results (in a three-dimensional sense) will indicate different values of repeatability in different regions of the work volume.

Measuring some form of repeatability is usually fairly simple. A nominal location in the workspace is selected for testing and a sensor-equiped fixture is located there. The robot is then placed in the fixture a number of times and the sensor data are recorded. The sensor may record one, two or three components of position together with other information enabling orientation information to be calculated. Such a fixture, which uses six LVDT position transducers, is shown in Figure 7.2, and enables both position and orientation data to be derived.

Similar devices of varying degrees of complexity are to be found in the literature and include the Ranky test aparatus [22], the IPA method [24], and the Ford Motor Company robot test station [23].

Based on the arguments of the previous paragraphs, it is reasonable to ask where a fixture such as the one shown in Figure 7.2 should be placed to perform the repeatability tests. Mooring and Pack [18] define a "position index" that indicates the sensitivity of endpoint movement to each joint motion through a Jacobian formulation. It is then suggested that the "average" repeatability could be obtained by locating the measuring fixture in a region of workspace where each joint position index is close to its mid-range value.

As indicated earlier, and in previous chapters in this book, accuracy is difficult to measure due to the lack of a well-defined and accessable base coordinate frame for the manipulator. If, as in the case of the PUMA, this frame is defined by the

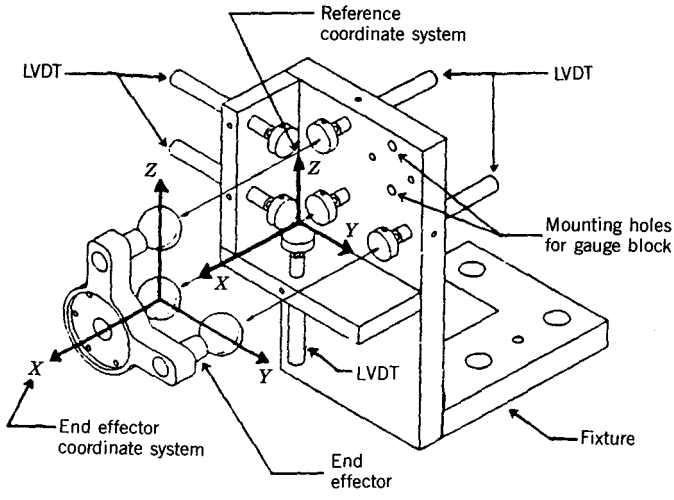


Figure 7.2. Repeatability fixture.

manufacturers to be inside the body of the arm, it is difficult to make measurements with respect to it. Usually the solution to the problem is to define an external and, therefore, measurable frame that ideally is fixed to the robot structure. In Chapter 6 dealing with the detailed case study the coordinate frame representing the base happens not to be fixed to the robot but coincident with the world frame of the system. In the ideal case in which the user robot world frame is attached to the manipulator structure, the fixed transformation between the user and manufacturers world frames would be identified as part of the calibration process as indicated in Chapters 2 and 4. Like the kinematic parameters, this transformation would be considered constant unless disturbed by collisions or dismantling the robot structure. Unless an external reference frame is defined and identified, absolute accuracy measures are meaningless. Here we define *absolute accuracy* to be accuracy measured with respect to the robots (user) base frame. Another measure of accuracy sometimes used is *relative accuracy* [5], which is accuracy measured with respect to a frame other than the user base frame. As such, relative accuracy may be assessed for robots that do not have an external user frame. These definitions are graphically shown in Figure 7.3.

Similar to measures of repeatability, accuracy may be defined in terms of one, two, or three position coordinates and/or orientation depending on the sophistication of the measuring system. If, as in the instance of the case study, we are able to measure both position and orientation, we would command the manipulator to move to a defined pose measured with respect to the manufacturers base frame. Knowing the transformation between the manufacturer and user base frames obtained through the calibration process, the pose with respect to the user frame may be calculated, and checked directly with the measuring system. Relative accuracy measures may be made in a similar fashion except that mesurements

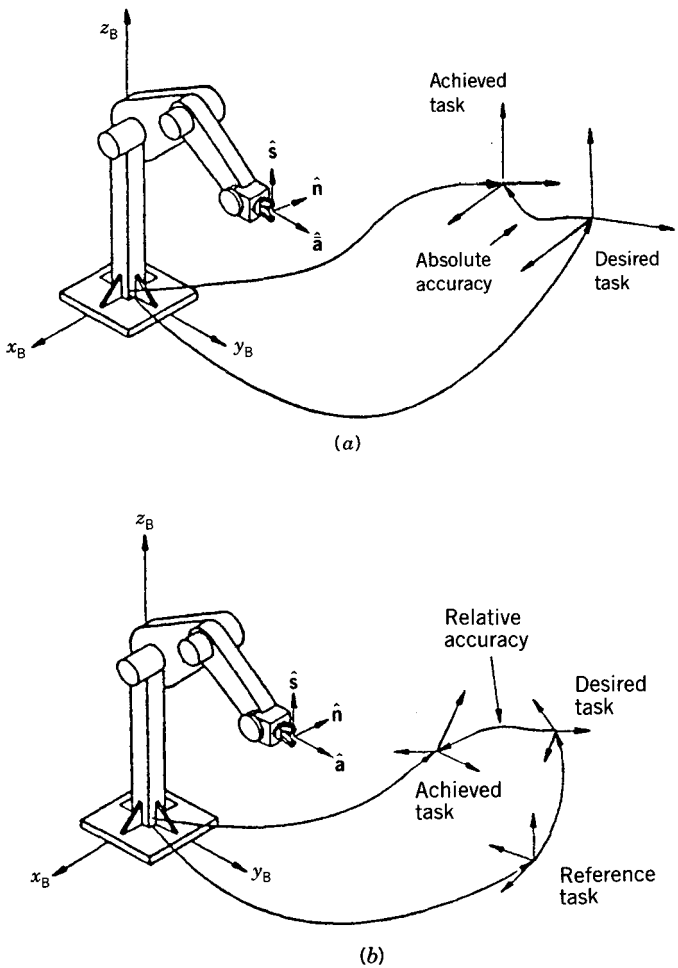


Figure 7.3. Definition of (*a*) absolute and (*b*) relative accuracy. Reprinted with permission of the authors [5].

with respect to the task frame, rather than the user base frame, are made. An example of relative accuracy assessment is given by Pathre [20] and uses a simple one-dimensional test fixture shown in Figure 7.4. The manipulator is commanded to move a given distance between the angle brackets. The exact distance is then measured with the dial gauges and compared with the commanded distance. Use of the fixture indicated that the relative accuracy was of the same order of magnitude as others had measured the absolute accuracy to be for the robot under study.

In some instances relative accuracy may be measured with respect to a precision template. These techniques have been widely used for testing coordinate measuring machines since these have integral position read-out data,

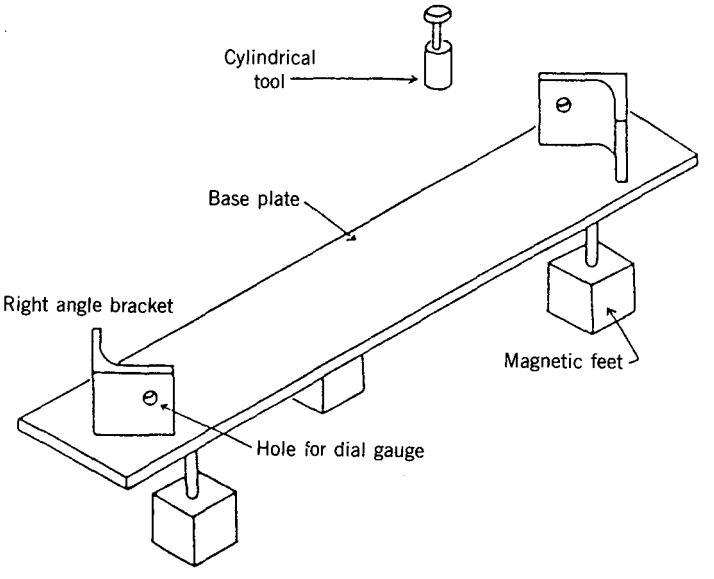


Figure 7.4. Fixture used to measure relative accuracy. Reprinted with permission of the author [20].

and orientation performance is rarely required. Examples of this technique include the calculation of the spacing of holes in a bar [3], and the ability of the machine to follow an exact circular template place in different orientations [9, 10]. These tests are interesting since the orthogonality of the machines axes, the location and orientation of the test piece, and the shape of the circular path traced by the machine allow the principal sources of kinematic error to be identified directly from the test results without resorting to an identification procedure. It is assumed that the shape of the template is known to at least one order of magnitude better than the required precision of the calibration process itself. Readers are encouraged to review other works dealing explicitly with measurement methods [13] that may be used both for static and dynamic position determination.

7.3 PATH CONTROL

Sometimes referred to as dynamic accuracy, this measure indicates the ability of a manipulator to follow a predetermined path. All of the performance qualifiers indicated at the beginning of the chapter will influence the actual path of the manipulator, together with other performance primitives such as payload, compliance, and, in particular, speed. Again, few standards are available (RIA R15.05 deals exclusively with static performance), although French researchers [15] have proposed measurement methods. Typical industrial tests usually require

the robot to follow a planar rectangular path or follow a circular trajectory with the performance measure being calculated in the form of a relative accuracy (deviation from ideal) as a function of speed and/or payload. Examples include the Ford Motor Company tests [26] where a rectangular path and a circular path are traced in each of three orthogonal planes at 10%, 50%, and 100% rated speed, maximum rated payload with the machine warmed up. Specific tests for Scara machines have also been proposed [16].

Precise, dynamic measurement of robot end effector position is very challenging. Measurement techniques for assessing dynamic accuracy include direct tracing of paths by a pen held by the robot gripper, although better methods incluce the use of fixed cameras [11, 16] or high-precision, high-speed tracking devices [6, 14] capable of following true three-dimensional motion. Other methods for measuring path accuracy [16] include the specification of the boundary width for a single linear movement.

7.4 SPEED

Speed may be specified in several ways. The most common method used by manufacturers [1] is to provide the maximum joint velocities, since this requires knowledge only of the servo motor characteristics and the gear drive. Some specifications may also provide information on the maximum speed of the tool. This requires computation of the forward kinematics, or Jacobian for instantaneous velocities, and will again be a function of the location of the trajectory in the workspace. Some manipulator manufacturers argue correctly that cycle time is a better measure of speed than simple end point velocity, since it takes into account the acceleration and deceleration phases of the motion. Although this test will provide information on the effectiveness of the control system on the transient portion of the motion, the path over which the cycle time is measured is difficult to define since it may bias the results for manipulators with certain kinematic configurations. A further refinement would be to specify a path that was unnatural to the robot, so that the computational efficiency of the controller could be assessed. Examples of such paths would be circular in nature for Cartesian configurations, or straight line motion for cylindrical machines. Again any measured speed should be qualified by the factors previously mentioned and other performance primitives, perhaps the most important of which would be payload, closeness to singularities, and workspace location.

7.5 PAYLOAD

Representative manipulator payload to weight ratios are in the region of 5–10%, which is relatively poor compared to the humans 50% figure. This can be attributed to the fact that for a fixed compliance (see later section) and a maximum end point error, the maximum allowable payload is a linear function of the manipulator characteristic length, r, which may be thought of as the moment

arm from the payload to the base. The mass of the manipulator arm, however, will vary as r^3. Humans outperform machines in this area due to their ability to vary compliance automatically with the task they are performing.

The ability of the joint servos and the manipulator structure to support a payload depends on the instantaneous kinematic configuration [1], and the speed with which the manipulator is moving. These factors, together with the repeatability and accuracy (static and dynamic), are related through complex, nonlinear equations. Any quoted payload figure must be assumed to represent the value consistent with the other performance measures provided by the manufacturer such as expected repeatability within a defined workspace. Exceeding the payload figure may, at best, lead to poor static and dynamic performance and at worst damage the drive components due to high acceleration/deceleration of the arm. It must be remembered that payload is anything attached to the end point of the manipulator, and includes the weight of the end effector and inertial loading.

7.6 THERMAL SENSITIVITY

The effects of temperature on manipulator performance may be subdivided into two mechanisms: the initial warm-up period of the device and the subsequent environmental changes following warm-up. Although manufacturers should specify the warm-up time for their products, the user may consider performing a repeatability test from a cold start. The actual warm-up period will then be the time required for repeatability measure to stabilize, since during the warm-up phase considerable drift in the global repeatability measure will be observed. Such a test was performed by Mooring et al. [17] with a hydraulic robot. Since the temperature of the working fluid changes rapidly from a cold start, hydraulic machines tend to exhibit stronger thermal sensitivity than electric drives and the proximity of hoses to the structure of the machines may produce local thermal expansion phenomena. Mooring et al. [17] found that the repeatability changed significantly during the warm-up period of about 20 min. Such tests are, in fact, measures of robot accuracy since variations relative to the world frame are measured.

The problem is not limited to hydraulic drive machines. Stauffer [23] indicated that with a particular electric drive machine, repeatability errors were two to five times the stated values, and other performance specifications could not be met until 10–12 min had elapsed from initial start-up. In addition, performance was adversely affected when the robot was inactive for 15 min, and if the machine was shut down for more than 30 min, it was then considered cold. Most of these problems were found to be caused by friction, particularly at the joint seals. It is well known that friction, and stiction in particular, are very sensitive to temperature.

Once the manipulator system has warmed up, temperature changes in the operating environment can produce significant positional errors. As indicated by Lau et al. [14], Juberts [8] measured a 0.0015 in. change in repeatability when

the room temperature changed by 7°F. Stauffer [23], however, found that the intermittent operation of an air conditioning system changed robot positioning accuracy by 0.010 in., leaving the repeatability relatively unchanged.

Steady-state temperature effects may be calibrated in much the same way as kinematic perturbations, although it is usual to use only simplified, linear models, and is more commonly applied to coordinate measuring machines (CMMs) than industrial manipulators [3]. Zhang et al. [7] performed a compensation experiment on a CMM in which temperature and kinematic modeling were undertaken. Since they were principally interested in position compensation in the orthogonal X, Y, and Z directions, the thermal model takes the form of

$$\Delta X_T = \alpha_x X T \tag{7.1}$$

where ΔX_T is the positional correction in the X direction, X is the scale readout value, T is the temperature, and α_x is an experimentally measured thermal expansion coefficient. Similar expressions applied to the Y and Z directions, although different thermal expansion coefficients are needed because variations in stiffness, materials, and construction constraints produce different expansions for the same change in temperature. The value of α_x, for example, was calculated by experimentally measuring the scale error ΔX_T at different positions, X, for varying temperature T. An example of one of their experimental graphs is shown in Figure 7.5. They conclude that although more complex thermal models were

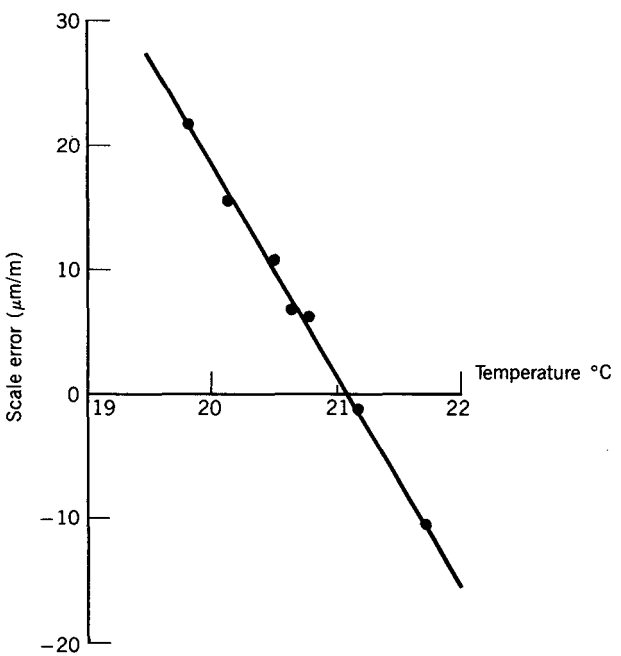


Figure 7.5. Effect of temperature on scale error.

investigated, there was negligible improvement over the linear model indicated above. Once identified, the thermal expansion coefficients may be used to calculate the positional corrections required for each axis, although real time measurement of temperature is required.

It may be seen that this process of thermal calibration is performed in the same manner as kinematic calibration by modeling (Equation 7.1), measurement (Figure 7.5), parameter identification (slope of line in Figure 7.5), and compensation (ΔX_T).

7.7 COMPLIANCE

Compliance is defined as the deflection of an elastic structure divided by the force causing the deflection, and is therefore the inverse of stiffness. The problem of manipulator compliance manifests itself when gravitational and payload forces displace the end point from the desired location thereby causing unwanted displacements of the tool. In general, there are three elements of a manipulator that may be regarded as compliant: the joint servo system, the joint transmission system, and the arm structure itself.

All joint servo systems have a natural stiffness since their basic feedback algorithm detects an angular displacement error and corrects this by applying a joint torque causing the error to diminish. In many manipulators this stiffness can be felt directly by manually displacing the joint a small amount and "feeling" the servo attempting to reduce the sensed error to zero. The error caused by joint compliance depends on the sophistication of the controller. It is known that a simple proportional controller, for example, has finite error for a constant load torque, such as gravity, whereas integral control eliminates this error.

Stiffness in the joint transmission systems leads to end point position errors irrespective of the type of joint control system employed. Figure 7.6 shows, for example, the major components of the drive system for the wrist roll axis of a PUMA manipulator.

It can be appreciated that since the angular position sensor (encoder) for this joint is located at the motor end of the transmission, a load torque might cause roll rotation of the tool to occur due to torsional compliance of the transmission components, particularly the connecting rod, without a rotation being detected by the encoder. Although the location of the drive motor remote from the wrist reduces in inertial loading on other drives, the errors due to transmission compliance are significant, and have ultimately led to the design of direct drive manipulators [23].

The drive components shown in Figure 7.6 are housed within the manipulator structure, which is usually a load bearing beam, or box-frame construction. Under gravitational loading this structure is also compliant, and may lead to unwanted positioning errors. Deformations will depend on the particular kinematic configuration of the manipulator as well as the design of the structure.

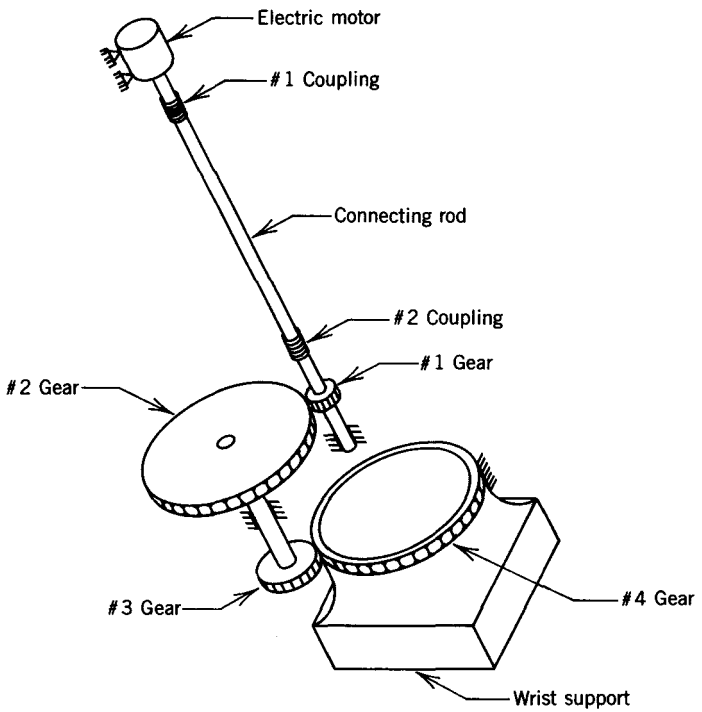


Figure 7.6. Wrist roll axis of PUMA manipulator. Adapted from reference [12]. Reprinted by permission of John Wiley & Sons, Inc.

Some research indicates experiments aimed at determining manipulator compliance and its significance on positioning errors. Whitney et al. [25] performed a static deflection test on a PUMA as shown in Figure 7.7

In this test, forces (weights) were applied at F_1 and F_2 and deflections were measured at locations 1–7. From these tests it was concluded that structural stiffness was negligible compared with joint stiffness, which could be modeled with a linear torsional spring. Mention was made of observed joint angle changes,

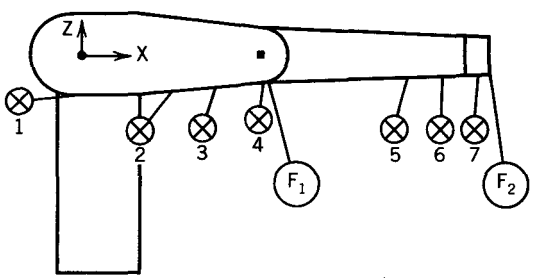


Figure 7.7. Deflection test on PUMA arm. Adapted from reference [12]. Reprinted by permission of John Wiley & Sons, Inc.

which indicated servo, as opposed to transmission, compliance. Whitney et al. further concluded that stiffness effects dominated errors caused by errors in kinematic parameters, which is in contrast to the findings of Mooring and Padavala [19]. It appears that in some manipulators, stiffness dominates kinematic errors while in others the reverse is true.

Compliance is certainly a problem in some industrial applications. In one known case, a large manipulator uses a variety of tools such as drills, countersink drills, and debur tools to process large aircraft structures. Although each tool operates sequentially on a fixed location, the difference in weight of each tool causes the manipulator to move to what appear to be separate positions. This must be remedied by reteaching the manipulator a different location depending on which tool is held by the robot. A similar example is the case of a robot being used for spot welding where the cable to the welder is heavy and stiff. Experience showed that simply rearranging the cable necessitated reteaching critical task points. Clearly these are cases for compliance compensation.

7.8 CONCLUSION

In this chapter we have attempted to demonstrate both the importance of good performance measurements and the difficulty of developing a standard for the evaluation of manipulator performance. Although a uniform testing standard for all manipulators would benefit both the users and manufacturers of robot manipulators, the complexity of such a general standard makes its development and universal acceptance a difficult task. Many companies, therefore, have resorted to the establishment of internal performance or acceptance standards that are tailored to their needs.

REFERENCES

[1] J. N. Anderson. Specifications. In *Encyclopedea of Robotics*, Vol. 3, pp. 1641–1657. Wiley Interscience, New York, 1988.

[2] ANSI/RIA. Evaluation of static performance characteristics for the comparison of manipulating industrial robotic systems. 1988. Draft Standard—R15.05 (Revision 10) Performance Subcommittee.

[3] W. Beckwith. Private communication, 1985.

[4] K. Busch, H. Kunzmann, and F. Walele. Calibration of coordinate measuring machines. *Precision Engineering*, 7(3):139–144, July 1985.

[5] J. Colson and N. D. Perreira. Robot system pose performance: Definitions and analysis. In *Proceedings of the ASME International Computers in Engineering Conference*, Boston, Massachusetts, August 1985.

[6] J. Gilby and G. Parker. Laser tracking system to measure robot arm performance. *Sensor Review*, 2(4):180–184, October 1892.

[7] G. Zhang, R. Veale, T. Chalton, B. Borchardt, and R. Hocken. Error compensation of coordinate measuring machines. *Annals of the CIRP*, 34(1):445–448, 1985

[8] M. Juberts. Repeatability measurements of a vision servoed manipulator using an optoelectronic remote 3-d tracking system. In *Proceedings of IEEE Conference on Systems, Man, and Cybernetics*, Tuscon, Arizona, 1985.

[9] W. Knapp. Circular test for three axis coordinate measuring machines and machine tools. *Precision Engineering*, 5(3):115–124, 1983.

[10] W. Knapp. Test of the dimensional uncertainty of machine tools and measuring machines and its relation to the machine errors. *Annals of CIRP*, 32(1):459–464, 1983.

[11] A. Kochan. Institute develops robot evaluation tests. *Assembly Automation*, 72–74, May 1986.

[12] K. Lau, N. Dagalakis, and D. Myers. Testing. In *Encyclopedea of Robotics*, Vol. 3, pp. 1753–1769. Wiley Interscience, New York, 1988.

[13] K. Lau and R. Hocken. A survey of current robot metrology methods. *Annals CIRP*, 33(2):485–488, 1984.

[14] K. Lau, R. Hocken, and L. Haynes. Robot end point sensing using a laser tracking system. In *Proceedings of Workshop on Robot Standards*, pp. 104–111, Detroit, Michigan, June 1985.

[15] A. Liegeois. *Robot Technology Volume 7: Performance and Computer Aided Design*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[16] H. Makino. Standard performance tests for planar positioning assembly robots. *Annals CIRP*, 34(1):33–36, 1985.

[17] B. W. Mooring, D. N. Bingham, and J. Hoffman. Enhancement of repeatability during warm-up for the IBM 7565 robots. In *Proceedings of the ASME International Computers in Engineering Conference*, pp. 259–262, Boston, Massachusetts, 1985.

[18] B. W. Mooring and T. J. Pack. Aspects of robot repeatability. *Robotica*, 5:223–230, 1987.

[19] B. W. Mooring and S. S. Padavala. The effect of model complexity on robot accuracy. In *Proceedings of 1989 IEEE Conference on Robotics and Automation*, pp. 593–598, IEEE, May 1989.

[20] U. Pathre. *Robot Accuracy Calibration and Compensation.* Master's thesis, University of Rhode Island, Mechanical Engineering Department, 1986.

[21] J. Prange and J. Peyton. Standards. In *Encyclopedea of Robotics*, Vol. 3, pp. 1663–1667. Wiley Interscience, New York, 1988.

[22] P. G. Ranky and C. Y. Ho. *Robot Modeling, Control and Software with Applications.* IFS Publications Ltd., Bedford, U.K., 1988.

[23] R. Stauffer. Robot accuracy. *Robotics Today*, 43–49, April 1985.

[24] H. J. Warnecke. Test stand for industrial robots. In *Proceedings of the 7th International Symposium on Industrial Robots*, Stuttgart, October 1977.

[25] D. E. Whitney, C. A. Lozinski, and J. M. Rourke. Industrial robot forward calibration method and results. *Journal of Dynamic Systems, Measurement, and Control*, 108(1):1–8, March 1986.

[26] M. Wodzinski. Putting robots to the test. *Robotics Today*, 17–21, June 1987.

APPENDIX A

PARAMETER IDENTIFICATION PROGRAM

```
      Implicit Real*8 (a-h,o-z)
      Real*8 Tew(60,4,4), Joint(60,6)
      Real*8 Parm(4), x(30), f(360), xjac(360,30), xjtj(465)
      Real*8 work(1335)
      Character Fname*15

      External PIndex

      Common/ben1/ Tew, Joint, Npose, Niter

      Niter = 0

      write(*,'(A\)') ' Please Input the Data File Name --> '
      read(*,'(A15)') Fname
      open(10, file = Fname, status = 'OLD')

      Do 2 i = 1, 60
           Read(10,*,end = 3) (Joint(i,j),j=1,6)
           Read(10,*) (Tew(i,1,k), k = 1, 4)
           Read(10,*) (Tew(i,2,k), k = 1, 4)
           Read(10,*) (Tew(i,3,k), k = 1, 4)
           Tew(i,4,1) = 0.0d0
           Tew(i,4,2) = 0.0d0
           Tew(i,4,3) = 0.0d0
           Tew(i,4,4) = 1.0d0
2     Continue
3     Npose = i - 1

      m     = npose * 6
      n     = 30
      nsig  = 4
      eps   = 0.0d0
      delta = 0.0d0
      maxfn = 1000
      iopt  = 0
      ixjac = 360

      parm(1) = 0.0d0
      parm(2) = 0.0d0
      parm(3) = 0.0d0
      parm(4) = 0.0d0
```

```
      Do 4 i = 1, 30
          x(i) = 0.0d0
4     Continue

      Call zxssq(PIndex, m, n, nsig, eps, delta, maxfn, iopt, parm,
     &           x, ssq, f, xjac, ixjac, xjtj, work, infer, ier)

      Write(*,'(//,'' IER = '',I5,//,'' SSQ = '',E15.5)') ier, ssq

      Do 5 i = 1, 30
         write(*,'('' x('',I2,'') = '',F25.6)') i, x(i)
5     Continue

      Open(10,File = 'NewPar.dat',status='New')

C     *** Write Out New Data File ***

      Do 6 i = 1, 30
         write(10,'('' x('',I2,'') = '',F25.6)') i, x(i)
6     Continue


      Close(10)

      Stop
      End

      Subroutine PIndex(x, m, n, f)

      Implicit Real*8 (a-h,o-z)
      Real*8 x(n), f(m), Param(7,6), Offset(6)
      Real*8 Tew(60,4,4), Joint(60,6)
      Real*8 Tr(4,4), Tm(4,4), Tri(4,4), Delta(4,4)
      Real*8 Theta(6)

      Common/ben1/ Tew, Joint, Npose, Niter

      Pi = Datan(1.0d0) * 4.0d0
      Dr = Pi / 180.0d0

      Call Par ( x, Param, Offset )

      Do 300 icnt = 1, Npose

         Do 3 i = 1, 6
            theta(i) = Joint(icnt,i) + Offset(i)
3        Continue

         Call Forward( Param, theta, Tr)

         Call Matinv( Tr, Tri)

         Do 4 i = 1, 4
            Do 4 j = 1, 4
               Tm(i,j) = Tew(icnt, i, j)
4        Continue

         Call Mamult(Tri, Tm, Delta)

      index = (icnt - 1) * 6

      f(index + 1) =  Delta(1,2)
      f(index + 2) =  Delta(1,3)
      f(index + 3) =  Delta(2,3)
      f(index + 4) =  Delta(1,4)
      f(index + 5) =  Delta(2,4)
      f(index + 6) =  Delta(3,4)
```

```
300     Continue

        niter = niter + 1
        write(*,'(//,'' Iteration Number '',I5)') niter
        write(*,'('' '',6E13.3)') x(1), x(2), x(3), x(4), x(5), x(6)
        write(*,'('' '',6E13.3)') x(7), x(8), x(9), x(10), x(11), x(12)
        write(*,'('' '',6E13.3)') x(13),x(14),x(15),x(16),x(17),x(18)
        write(*,'('' '',6E13.3)') x(19),x(20),x(21),x(22),x(23),x(24)
        write(*,'('' '',6E13.3)') x(25),x(26),x(27),x(28),x(29),x(30)

        Return
        End
```

APPENDIX B

SUBROUTINES ASSOCIATED WITH MODIFIED DENAVIT–HARTENBERG METHOD

```
      Subroutine Par(x, Param, Offset )
      Real*8 x(30), Param(7,6), Offset(6)

      Param(1,1) =   32.878d0 + x(1)
      Param(1,2) =  -15.591d0 + x(2)
      Param(1,3) =  -90.000d0 + x(3)
      Param(1,4) =    0.000d0 + x(4)
      Param(2,1) =   14.681d0 + x(5)
      Param(2,2) =    0.000d0 + x(6)
      Param(2,3) =  -90.000d0 + x(7)
      Param(2,4) =    0.000d0 + x(8)
      Param(3,1) =   17.000d0 + x(9)
      Param(3,2) =    0.000d0 + x(10)
      Param(3,3) =    0.000d0 + x(11)
      Param(3,4) =    0.000d0 + x(12)
      Param(4,1) =    5.870d0 + x(13)
      Param(4,2) =   -0.800d0 + x(14)
      Param(4,3) =   90.000d0 + x(15)
      Param(4,4) =    0.000d0 + x(16)
      Param(5,1) =   17.050d0 + x(17)
      Param(5,2) =    0.000d0 + x(18)
      Param(5,3) =  -90.000d0 + x(19)
      Param(5,4) =    0.000d0 + x(20)
      Param(6,1) =    0.000d0 + x(21)
      Param(6,2) =    0.000d0 + x(22)
      Param(6,3) =   90.000d0 + x(23)
      Param(6,4) =    0.000d0 + x(24)
      Param(7,1) =    0.000d0 + x(25)
      Param(7,2) =    0.000d0 + x(26)
      Param(7,3) =    2.213d0 + x(27)
      Param(7,4) =    0.000d0 + x(28)
      Param(7,5) =    0.000d0 + x(29)
      Param(7,6) =   90.000d0 + x(30)

      Do 1 i = 1, 6
          Offset(i) = 0.0d0
1     Continue

      Return
      End
```

```
      Subroutine Forward( param, theta, T )

      Implicit Real*8 (a-h,o-z)
      Real*8 param(7,6), theta(6), T(4,4), l
      Real*8 A0(4,4), A1(4,4), A2(4,4)

      Do 1 i = 1, 4
         Do 2 j = 1, 4
            A0(i,j) = 0.0d0
2        Continue
         A0(i,i) = 1.0d0
1     Continue

      Do 10 jt = 1, 7

         If ((jt.ne.3).and.(jt.ne.7)) then

              r      = param(jt,1)
              l      = param(jt,2)
              alpha  = param(jt,3)
              If ( jt .eq. 1 ) then
                   th = 0.0d0 + param(1,4)
              else
                   th = theta(jt-1) + param(jt,4)
              endif

              Call Harden( r, l, alpha, th, A1 )

              Go to 100

         Endif

         If ( jt.eq.3 ) then

              l      = param(jt,1)
              alpha  = param(jt,2)
              beta   = param(jt,3)
              th  = theta(jt-1) + param(jt,4)

              Call Modden( l, alpha, th, beta, A1 )

              Go to 100

         Endif

         If ( jt.eq.7 ) then

              x  = param(jt,1)
              y  = param(jt,2)
              z  = param(jt,3)
              tx = param(jt,4)
              ty = param(jt,5)
              tz = theta(jt-1) + param(jt,6)

              Call Trans( x, y, z, tx, ty, tz, A1 )

              Go to 100

         Endif

100      Call Mamult( A0, A1, A2 )

         Do 101 i = 1, 4
            Do 102 j = 1, 4
               A0(i,j) = A2(i,j)
102         Continue
101      Continue

10    Continue
```

```
      Do 201 i = 1, 4
         Do 202 j = 1, 4
            T(i,j) = A0(i,j)
202      Continue
201   Continue

      Return
      End

      Subroutine Harden( r, l, alpha, theta, A )

      Implicit Real*8 (a-h,o-z)
      Real*8 l, A(4,4)

      Pi = 4.0d0 * datan(1.0d0)
      Dr = pi/ 180.0d0

      sa = dsin(alpha*Dr)
      ca = dcos(alpha*Dr)
      st = dsin(theta*Dr)
      ct = dcos(theta*Dr)

      A(1,1) =  ct
      A(1,2) = -st*ca
      A(1,3) =  st*sa
      A(1,4) =  l*ct
      A(2,1) =  st
      A(2,2) =  ct*ca
      A(2,3) = -ct*sa
      A(2,4) =  l*st
      A(3,1) =  0.0d0
      A(3,2) =  sa
      A(3,3) =  ca
      A(3,4) =  r
      A(4,1) = 0.0d0
      A(4,2) = 0.0d0
      A(4,3) = 0.0d0
      A(4,4) = 1.0d0

      Return
      End

      Subroutine Modden( l, alpha, theta, beta, A )

      Implicit Real*8 (a-h,o-z)
      Real*8 l, A(4,4)

      Pi = 4.0d0 * datan(1.0d0)
      Dr = pi/ 180.0d0

      sa = dsin(alpha*Dr)
      ca = dcos(alpha*Dr)
      sb = dsin(beta*Dr)
      cb = dcos(beta*Dr)
      st = dsin(theta*Dr)
      ct = dcos(theta*Dr)

      A(1,1) = -sa*sb*st + cb*ct
      A(1,2) = -ca*st
      A(1,3) =  sa*cb*st + sb*ct
      A(1,4) =  l*ct
      A(2,1) =  sa*sb*ct + cb*st
      A(2,2) =  ca*ct
      A(2,3) = -sa*cb*ct + sb*st
      A(2,4) =  l*st
      A(3,1) = -ca*sb
      A(3,2) =  sa
      A(3,3) =  ca*cb
      A(3,4) = 0.0d0
```

```
A(4,1) = 0.0d0
A(4,2) = 0.0d0
A(4,3) = 0.0d0
A(4,4) = 1.0d0

Return
End

Subroutine Trans( x, y, z, tx, ty, tz, A )

Implicit Real*8 (a-h, o-z)
Real*8 A(4,4)

Pi = 4.0d0 * datan(1.0d0)
Dr = pi / 180.0d0

sx = dsin(tx * Dr)
cx = dcos(tx * Dr)
sy = dsin(ty * Dr)
cy = dcos(ty * Dr)
sz = dsin(tz * Dr)
cz = dcos(tz * Dr)

A(1,1) =  CY*CZ
A(1,2) =  -CX*SZ + SX*SY*CZ
A(1,3) =  CX*SY*CZ + SX*SZ
A(1,4) =  -Y*CX*SZ + Y*SX*SY*CZ + CX*SY*CZ*Z + SX*SZ*Z + CY*CZ*X
A(2,1) =  CY*SZ
A(2,2) =  CX*CZ + SX*SY*SZ
A(2,3) = CX*SY*SZ - SX*CZ
A(2,4) = Y*CX*CZ + Y*SX*SY*SZ + CX*SY*SZ*Z - SX*CZ*Z + CY*SZ*X
A(3,1) =  -SY
A(3,2) = SX*CY
A(3,3) = CX*CY
A(3,4) = Y*SX*CY + CX*CY*Z - SY*X
A(4,1) = 0.0d0
A(4,2) = 0.0d0
A(4,3) = 0.0d0
A(4,4) = 1.0d0

Return
End
```

```
      SUBROUTINE VECROT(U,P,THETA,A)
C
C -----------------------------------------------------------------------
C
C     This subroutine computes a homogeneous transformation matrix
C     based on the point-unit vector approach. The vector U is dimensioned
C     U(3) and represents the unit vector about which the body will rotate.
C     The vector p is dimensioned P(3) also and it contains the coordinates
C     of a point through which the rotation axis passes. THETA is the angle
C     of rotation about U. The resulting matrix A is the 4x4 transformation
C     matrix that is passed back to the calling program. All arguments are
C     double precision.
C
C -----------------------------------------------------------------------
C
      REAL*8 U(3),P(3),THETA,A(4,4),ST,CT,VT,DR
C
C     Define constants PI, ST, CT, and VT
C
      DR=4.0D0 * DATAN(1.0D0) / 180.0D0
      ST= DSIN(THETA*DR)
      CT= DCOS(THETA*DR)
      VT= 1.0D0-CT
C
C     Assign values in matrix
C
      A(1,1) = U(1) * U(1) * VT + CT
      A(1,2) = U(1) * U(2) * VT - U(3) * ST
      A(1,3) = U(1) * U(3) * VT + U(2) * ST
      A(2,1) = U(2) * U(1) * VT + U(3) * ST
      A(2,2) = U(2) * U(2) * VT + CT
      A(2,3) = U(2) * U(3) * VT - U(1) * ST
      A(3,1) = U(3) * U(1) * VT - U(2) * ST
      A(3,2) = U(3) * U(2) * VT + U(1) * ST
      A(3,3) = U(3) * U(3) * VT + CT
      A(4,1)=0.0D0
      A(4,2)=0.0D0
      A(4,3)=0.0D0
      A(4,4)=1.0D0
      A(1,4)=(1.0D0 - A(1,1)) * P(1) - A(1,2) * P(2) - A(1,3) * P(3)
      A(2,4)=-A(2,1) * P(1) + (1.0D0 - A(2,2)) * P(2) - A(2,3) * P(3)
      A(3,4)=-A(3,1) * P(1) - A(3,2) * P(2) + (1.0D0 - A(3,3)) * P(3)
      RETURN
      END
```

APPENDIX D

GENERAL PURPOSE SUBROUTINES

```
        SUBROUTINE MAMULT(A1,A2,A3)
C
C ---------------------------------------------------------------------
C
C     This subroutine multiplies two 4x4, double precision matrices
C     (A1 and A2). The resulting product is the double precision
C     matrix A3.
C
C ---------------------------------------------------------------------
C
        REAL*8 A1(4,4),A2(4,4),A3(4,4)
        DO 100 I=1,4
        DO 100 J=1,4
        A3(I,J)=0.0D0
        DO 100 K=1,4
100     A3(I,J)=A3(I,J)+A1(I,K)*A2(K,J)
        RETURN
        END


        SUBROUTINE MATINV(A1,A1INV)
C
C ---------------------------------------------------------------------
C
C     This subroutine determines the inverse of a 4x4 double precision
C     homogeneous transformation matrix, A1. It returns the inverse in
C     the double precision matrix A1INV.
C
C ---------------------------------------------------------------------
C
        REAL*8 A1(4,4),A1INV(4,4)

        A1INV(4,1) = 0.0D0
        A1INV(4,2) = 0.0D0
        A1INV(4,3) = 0.0D0
        A1INV(4,4) = 1.0D0

        DO 100 I=1,3
            DO 100 J=1,3
                A1INV(I,J) = A1(J,I)
```

```
100     CONTINUE

        A1INV(1,4)=-(A1(1,1)*A1(1,4)+A1(2,1)*A1(2,4)+A1(3,1)*A1(3,4))
        A1INV(2,4)=-(A1(1,2)*A1(1,4)+A1(2,2)*A1(2,4)+A1(3,2)*A1(3,4))
        A1INV(3,4)=-(A1(1,3)*A1(1,4)+A1(2,3)*A1(2,4)+A1(3,3)*A1(3,4))

        RETURN
        END
```

INDEX

Accuracy:
  calibration, 140, 149, 154, 162, 167, 176, 204, 210, 242, 295
  experimental, 11, 295
  relative, 303
  robot, 5, 11, 13, 15, 32, 107, 108, 196, 199, 207, 211, 216, 221, 225, 229, 242, 243, 297, 299, 301, 307
  sensor, 71, 80, 81, 87, 100, 108, 140, 148, 207, 275
Accuracy compensation:
  model based, 209, 215, 221, 225
  nongeometric, 214, 242, 253, 311
Albus, J. S., 253, 256, 264
Arm signature, 175, 176, 182
Assembly:
  circuit card, 13
  robot joint, 16

Barker, L. K., 166, 204, 205
Base orientation parameters, 203
Base transformations, 3, 39, 109, 167, 184, 192, 203, 273, 302

Calibration:
  forward and inverse, 208
  implementation, 18, 20, 207, 209, 215, 221, 225, 242, 264
  kinematic model selection, 19, 23, 146, 290
  levels, 17
  measurement configurations, 19, 71, 131, 140, 166, 175, 201, 281
  open research issues, 111, 140, 154, 264
  process of, 18
Calibration systems:
  design considerations, 6, 107, 140, 167, 225, 276
  fixtures, 6, 71, 87, 102, 109, 112, 161, 166, 275, 302
  tools, 6, 167, 176, 276
Camera (for measurement), 86
Case study, 266
Cerebellar model articulation controller (CMAC), 243, 253, 264
Chebyshev polynomials, 248
Circle point analysis (CPA), 166, 175, 182
Closed loop manipulators, 44, 56, 103
Compensation:
  geometric errors, 215, 221, 225
  nongeometric errors, 242, 264
Completeness, 19, 41, 66, 145, 201
Compliance, 5, 18, 25, 299, 309
Computational complexity, 218, 221, 239
Condition number, 140, 143, 146, 150, 153, 204
Coordinate measuring machine (CMM), 6, 71, 83, 92, 101, 104, 267, 275, 281, 295, 308
Convergence:
  compensation, 220
  identification, 114, 129, 131, 132, 135, 138, 140, 181

Data driven tasks, 209
Denavit-Hartenberg (DH):
  actual link parameter extraction, 108, 182, 185, 186
  error parameters, 39, 44, 110, 118, 145, 199, 229, 234, 240
  limitations, 38
  modeling convention, 23, 25, 26, 29, 67
  modified models, 41, 44, 182, 199, 222, 266, 288, 318
  parameter identification, 107, 133, 145
  PUMA 560 parameters, 219, 224, 240
Displacement transducers, short range, 87
Drive system nonlinearities, 17, 25, 113, 309

Encoder:
  absolute, 75
  incremental, 24, 74
  tachometer, 74
End effector, 2, 277
Error:
  calibration, 99, 108, 140, 162, 204
  differential transformations, 34, 110, 192, 193, 217, 278
  identification, 107, 108, 169, 176, 204, 225, 242
  linearized model, 32, 108, 155, 192, 196, 225, 230
  modeling, in Kalman filters, 131, 158
  nongeometric, 17, 18, 112, 113, 140, 209, 225, 229, 307, 309
  parametric uncertainty, 15, 32, 107, 109, 119, 121, 130, 163
  probabilistic models, 15, 107, 109, 119, 121, 130, 163
  significance, 15, 17
  sources, 15, 17, 32
  unmodelled, 140, 225, 290
  variance, 120, 128, 159, 163, 169
Estimation:
  direct search, 132
  Gauss-Newton, 138
  gradient methods, 135, 282
  Kalman filtering, 124, 157, 163, 204
  Levenberg-Marquardt (LM), 139, 204, 282, 296
  linear least squares, 114, 115, 153, 204
  linear regression, 166, 167
  maximum likelihood, 122, 125
  minimum variance, 119, 125
  Newton's method, 136
  nonlinear least squares, 132, 135, 153, 204, 282
  prediction and smoothing, 127
  state estimation, 124, 157
  unbiased, 118, 120, 126
Everett, L. J., 41, 57, 67
Experimental results, 6, 266

Flexible link, 5, 66, 309
Flexible manufacturing, 1, 13
Friction, 18, 307

Generalization, 254
Gravitational effects, 309

Hayati, S. A., 45, 48, 67, 102, 105, 199, 205, 208, 264
Hessian, 117, 136
Higher pair joints, 64
Hollerbach, J. M., 48, 67, 103, 105, 242, 265
Hydraulic robot, 307

Identification:
  base or tool parameters, 109, 146, 167, 192
  convergence, 114, 129, 131, 132, 135, 138, 140, 181, 283, 286
  error variance, 121, 129, 163, 169
  example, 144
  geometric, 108, 147, 165, 187
  kinematic parameters, 20, 106, 108, 182, 192, 282, 315
  parametric, 108, 163, 192, 309, 315
  recursive, 107, 125, 157, 163, 287
  speed of processing, 147, 153, 284
  stochastic, 107, 119, 124, 163
Inaccuracy sources, 15
International Mathematical and Statistical Library (IMSL), 122, 146, 205, 283, 316
Initialization, 25, 75, 78, 83, 94, 97
Inverse kinematics:
  analytic solution, 182, 207, 208
  numerical solution, 21, 208, 215, 253

Jacobian identification, 38, 108, 110, 123, 132, 136, 140, 143, 150, 153, 157, 192, 198, 201, 203, 222, 284
Jacobian inverse, 215, 220, 264
Jacobian manipulator, 161, 215, 216, 302
Joint:
  axis geometric features, 16, 166, 180, 182, 183
  axis identification, 56, 108, 165, 177
  axis misalignment, 15, 16, 40, 130, 290
  clearance effects, 5, 16, 18, 131
  command updating, 209, 215, 215, 218, 220, 225
  displacement transducer, 24, 71, 113, 131, 140, 146, 149, 207, 209, 211, 309
  friction, 18, 307

higher pair, 44, 64
offset, 32, 39, 41, 55, 110, 113, 147, 160, 230, 269, 273, 289, 295
prismatic, 42, 64, 145, 166, 182, 184, 201, 233
range of motion, 5, 140, 146, 151, 167, 175, 181, 211, 225, 232, 233
space, 4
spherical, 61
wobble, 124, 130, 167

Kalman filter, 109, 124, 141, 157
Kinematic:
large parameter errors, 39, 132, 167, 221, 242
model construction, 23, 41, 66, 108, 165, 182
parameter errors, 16, 32, 43, 114, 132, 160, 192, 199, 229
parameter extraction, 182, 186
sensitivities, 222, 230
Kinematic models:
closed loop manipulator, 56–64
Denavit–Hartenberg (DH), 23, 25, 26, 29, 67
Hayati's model, 199, 266, 288, 290, 318
modified DH, 41, 44, 182, 199, 222, 266, 288, 318
review of models, 44–66
S-Model, 182
zero reference model, 44, 49, 266, 271, 289, 290, 322

Lagrange multipliers, 178
Laser interferometer, 71, 80, 81, 94, 104
Learning, 256
Least squares identification:
linear, 96, 108, 111, 114, 121, 141, 146, 153, 163, 166, 246
minimum variance, 119
nonlinear, 108, 111, 132, 146, 153, 316
minimum variance, 125
regression, 166, 167
singular cases, 119
Level 1 calibration, 18, 24, 66
Level 2 calibration, 18, 25, 66
Level 3 calibration, 18
Levenberg–Marquardt (LM), 139, 204, 206, 282, 296
Linearization:
error models, 32, 108, 132, 154, 193, 196, 225, 226, 233
measurement equations, 114, 154
Linear quadratic regulators, 226
Linear regression, 167
Linear variable differential transformer (LVDT), 80, 87, 103, 302
Line of visibility, 176
Loading effects, 16, 299, 306, 309
LSQRR (subroutine), 122

Magnetostrictive transducer, 79
Measurement:
configurations, 19, 71, 131, 140, 146, 157, 175, 201, 243, 281
cost, 19, 71, 92, 284
ill-conditioned, 119
noise, 19, 99, 109, 112, 118, 119, 122, 140, 145, 146, 149, 158, 163, 169, 180
number of, 19, 109, 130, 131, 140, 146, 150, 157, 163, 201, 278, 286
planning, 140, 146, 157, 192
Measurement devices:
acoustic transducers, 79, 85, 100, 165
camera based, 86, 176
coordinate measuring machines (CMM), 6, 71, 83, 92, 101, 104, 267, 275, 281, 295, 308
laser interferometer, 71, 80, 81, 94, 99, 104
short range devices, 87, 102
theodolite, 71, 80, 90, 97
Minimum move, 5, 211
Minimum variance estimation, 119
Model:
actual, 15, 108, 146, 160, 192
completeness, 19, 41, 66, 145, 201
closed loop manipulator, 56–64
Denavit–Hartenberg, 23, 25, 26, 29, 67
equivalence, 19, 44, 66
errors, 141, 192
Hayati's model, 199, 266, 288, 290, 318
modified DH, 41, 44, 182, 199, 222, 266, 288, 318
nominal, 15, 108, 146, 160, 165, 215
proportionality, 19, 44, 66, 196
residuals, 111, 114, 124, 126
S-Model, 182
zero reference model, 44, 49, 266, 271, 289, 290, 322

Near parallel axes, 38
Noise:
covariance, 109, 118, 122, 124, 158, 163, 169
error parameter uncertainty, 15, 107, 119, 124, 159
Gaussian, 122, 124, 169
joint position transducer, 113, 131, 140, 146, 149
measurement, 19, 109, 146, 149, 158, 163, 169
probabilistic model, 119, 169
process, 15, 107, 119, 124, 159
Nonparametric compensation, 242

Norm, 142, 157, 163, 211, 213, 218, 230
Numerical analysis, 141, 154

Observability:
  index, 140, 154, 157, 204
  of parameter errors, 140, 145, 153, 154, 159, 161, 192, 201, 203
Observation strategy, 19, 140, 146, 150, 153, 157, 164, 175, 192
Off-line:
  programming, 1, 12, 208
  simulation, 153, 205
Open research issues, 111, 140, 154, 204
Optimal:
  control problem, 229
  design of accuracy compensators, 225
  observation strategy, 140, 154, 164, 205
Optical (camera) techniques, 176

Parameter extraction, 182, 186
Parameter identification program, 315
Path control, 305
Paul, R. P., 26, 29, 36, 68, 182, 186, 193, 206, 217, 235, 265
Payload, 16, 299, 306, 309
Performance, 5, 298, 300
Point measurement, 92
Polynomial approximation functions, 243
Pose:
  definition, 2
  error, 70, 155, 199, 209, 221, 223, 243
  fixture, 103
  full pose measurement, 70, 101, 112, 245, 286
  partial pose measurement, 70, 155, 199, 209, 221, 227, 243
Potentiometers, 72
Precision, manipulator, 11
Programming of robots, 1, 11, 108, 264
Properties of a good model, 41
Proportionality, 19, 44, 66, 196
PUMA 560:
  accuracy, 6, 11, 17
  calibration, 20, 102, 266
  drive system nonlinearities, 17, 25, 113, 309
  Jacobian, 218
  joint command variables, 220, 225, 250
  kinematic error parameters, 17, 43, 201, 202, 240
  nominal parameters, 51, 219, 224, 240, 268, 272, 302
  repeatability, 9

Quantization effects, 5, 211, 253

Reduced order error model, 199, 203, 204, 235
Regression:
  circle fitting, 166, 172, 177, 178
  goodness-of-fit, 169, 172, 175, 182
  iterative solution, 173, 178, 181
  line fitting, 167, 174
  noise model, 169
  plane fitting, 166, 171, 178
Repeatability, 4, 8, 9, 13, 15, 109, 131, 210, 212, 278, 299, 301, 307
Reprogramming, 209, 214, 234
Residuals, 111, 114, 124, 126, 140
Resolution, 5, 13, 24, 73, 75, 77, 81, 100, 211, 299, 301
Resolver, 77
Robotic application:
  off-line programming, 12, 208, 209, 221
  taught, 11, 209, 221, 232, 264
Robot replacement, 108, 209

Scaling, 140, 226
SCARA, 14, 256
Sensitivity:
  of identification algorithms to modeling errors, 131, 141
  kinematic, 222, 230
  single joint model, 55
  thermal, 299, 307
Shamma, J. S., 208, 242, 250, 264, 265
Simulation results, 140, 144, 160, 240
Singular:
  identification Jacobian, 138, 155
  measurement coefficient matrix, 119
  robot configurations, 161, 204, 215, 216, 221, 225, 226, 231, 233, 242, 264, 299
  value decomposition (SVD), 154, 201, 205
  values, 155
Sonic transducer, 84
Speed of manipulator, 306
Sklar, M. E., 166, 182, 186, 206
S-Model, 182
Standards, 298, 300
Stone, H. W., 17, 56, 68, 100, 104, 166, 175, 177, 178, 180, 182, 206

Task space, 3
Temperature effects, 299, 307
Theodolite, 71, 80, 90, 97
Thermal sensitivity, 307
Tolerance:
  assembly, 16, 66, 108, 209
  fabrication of calibration fixtures, 112
  machining, 66, 108, 209
  robot joint axes, 18, 130

Tool:
  calibration tool, 167, 276
  offset, 141, 147, 153
  orientation parameters, 202
  transformation, 41, 109, 167, 192
Transformation:
  base, 109, 192
  homogeneous, 27
  tool, 109, 192, 202
Triangulation 95, 97

Vision system, 13
Vuskovic, M. I., 221, 224, 265

Waldron, K. J., 218, 219, 265
Warm-up, 307
Whitney, D. E., 15, 17, 22, 48, 69, 90, 104, 113, 206, 208, 222, 242, 250, 264, 265, 310, 312
Workspace feedback, 12, 13
World coordinate frame, 41
Wrist:
  center, 5, 208, 218
  observable parameters, 141
  singularity, 242
  wrist-tool distance, 154
Wu, C. H., 32, 48, 67, 92, 104, 206, 221, 265

Zero position, 32, 39, 41
Zero reference model, 49, 266, 271, 289, 322
Zhuang, H., 196, 225, 265
ZXSSQ (subroutine), 285